

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**DESIGN AND ANALYSIS
OF AN
OBJECT-ORIENTED DATABASE
OF
ELECTRONIC WARFARE DATA**

by

Kevin M. Coyne

March 1996

Thesis Advisor:
Co-Advisor:

David K. Hsiao
C. Thomas Wu

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 3

19960724 044

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time reviewing instructions, searching existing data sources gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE March 1996		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE Design and Analysis of an Object-Oriented Database of Electronic Warfare Data(U)				5. FUNDING NUMBERS
6. AUTHOR(S) Coyne, Kevin, M.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSORING/ MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 words) The Electronic Warfare Integrated Reprogramming Database (EWIRDB) is the primary Department of Defense (DoD) approved source of electronic warfare (EW) data. Its utilization in the areas of battle planning and EW research enables our military forces to effectively exploit the electromagnetic spectrum and shape the outcome of battle. The EWIRDB, however, lacks a viable conceptual data model. EWIRDB data are represented in disjoint parametric tree models that are implementation-oriented; to the extent that the tree structures are used as conceptual modeling tools, their hierarchical form is too restrictive to adequately describe EW data semantics. Moreover, these structures address only technical parametric data. Associated administrative, reference, and comment data are excluded. In practice, the EWIRDB is described in terms of the coded and record-based format of its output media, not its conceptual model. The primary goal of this thesis is the development of a semantically-improved conceptual design of EWIRDB data based on the object-oriented data model (OODM). The secondary goal of the thesis is the specification of a logical design, based on the new conceptual design, to provide the structure for a subsequent implementation of EWIRDB data on the Multimodel and Multilingual Database System (M ² DBS) in the Laboratory for Database Systems Research at the Naval Postgraduate School. The results of the work contained herein are: (1) an object-oriented conceptual design of EWIRDB data that supports the semantics of both the file format and tree structures, and (2) the specification of an object-oriented logical design for an M ² DBS implementation of sample EWIRDB data.				
14. SUBJECT TERMS Object-oriented Database Design Electronic Warfare Electronic Warfare Integrated Reprogramming Database				15. NUMBER OF PAGES 106
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified		18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified		19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified
				20. LIMITATION OF ABSTRACT UL

Approved for public release; distribution is unlimited

**DESIGN AND ANALYSIS
OF AN OBJECT-ORIENTED DATABASE
OF ELECTRONIC WARFARE DATA**

Kevin M. Coyne
Lieutenant, United States Navy
B.S., United States Naval Academy, 1987

Submitted in partial fulfillment of the
requirements for the degree of

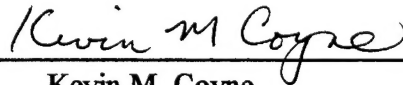
MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

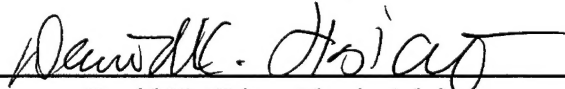
March 1996

Author:

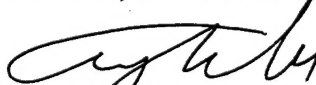


Kevin M. Coyne

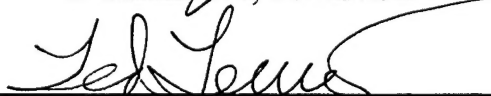
Approved by:



David K. Hsiao, Thesis Advisor



C. Thomas Wu, Co-Advisor



Ted Lewis, Chairman
Department of Computer Science

ABSTRACT

The Electronic Warfare Integrated Reprogramming Database (EWIRDB) is the primary Department of Defense (DoD) approved source of electronic warfare (EW) data. Its utilization in the areas of battle planning and EW research enables our military forces to effectively exploit the electromagnetic spectrum and shape the outcome of battle. The EWIRDB, however, lacks a viable conceptual data model. EWIRDB data are represented in disjoint parametric tree models that are implementation-oriented; to the extent that the tree structures are used as conceptual modeling tools, their hierarchical form is too restrictive to adequately describe EW data semantics. Moreover, these structures address only technical parametric data. Associated administrative, reference, and comment data are excluded. In practice, the EWIRDB is described in terms of the coded and record-based format of its output media, not its conceptual model.

The primary goal of this thesis is the development of a semantically-improved conceptual design of EWIRDB data based on the object-oriented data model (OODM). The secondary goal of the thesis is the specification of a logical design, based on the new conceptual design, to provide the structure for a subsequent implementation of EWIRDB data on the Multimodel and Multilingual Database System (M²DBS) in the Laboratory for Database Systems Research at the Naval Postgraduate School.

The results of the work contained herein are: (1) an object-oriented conceptual design of EWIRDB data that supports the semantics of both the file format and tree structures, and (2) the specification of an object-oriented logical design for an M²DBS implementation of sample EWIRDB data.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. AN OVERVIEW OF THE EWIRDB	1
B. THE FORMAT OF THE EWIRDB	4
1. The Parametric Tree Model	5
a. The Parametric Tree Structure and Notation	5
b. The Limitations Of Hierarchical Data Modeling	9
2. The File Structure of the Output Data	11
3. Summary	16
C. THESIS OBJECTIVES	17
D. THE ORGANIZATION OF THE THESIS	17
II. DESIGN CONCEPTS	19
A. THE CONCEPTUAL DESIGN	19
B. THE LOGICAL DESIGN	20
III. OBJECT-ORIENTED DESIGN CONCEPTS	23
A. OBJECTS	24
1. Object State	25
a. Simple Attributes	26
b. Complex Attributes and Relationships	26
2. Object Behavior and Encapsulation	27
B. OBJECT CLASSES	29
1. Instantiation and Classification	31
2. Generalization and Specialization	32
3. Aggregation	33
4. Covering	34
IV. A CONCEPTUAL OBJECT-ORIENTED EWIRDB	35
A. A GLOBAL SCHEMA	37

B. THE EMITTER SCHEMAS	42
1. The Kiting Emitter Class	43
2. The Association of an Emitter to its Signal	46
3. The S&TI and USNCSDB Emitter Classes	47
C. THE SCHEMAS WITHIN LOGICAL GROUPS.....	50
1. Antenna Data	50
2. Signal Data	52
3. Receiver Data	55
4. WARM Data	58
D. THE PARAMETRIC DATA CLASS	58
E. SUFFIX CODING AND THE SUFFIX TABLE	63
V. A LOGICAL OBJECT-ORIENTED EWIRDB	65
VI. CONCLUSION	83
LIST OF REFERENCES	87
INITIAL DISTRIBUTION LIST	89

LIST OF FIGURES

1. The Merging of Data into the EWIRDB	3
2. The Pulsed/Continuous Wave (P/CW) Parametric Tree	6
3. The Receiver Performance Assessment (RPA) Tree	6
4. A Detailed Portion of the P/CW Tree	8
5. A Description of TERF Elements	13
6. An Object Class and its Objects	30
7. Conceptual Object-Oriented Design Symbology	36
8. The Conceptual Schema: A Global View	38
9. The Conceptual Schema: Kilting Emitter Data	44
10. The Conceptual Schema: S&TI Emitter Data	48
11. The Conceptual Schema: USNCSDB Emitter Data	49
12. The Conceptual Schema: Antenna Data Enlargement	51
13. The Conceptual Schema: Signal Data Enlargement	53
14. The Conceptual Schema: Receiver Data Enlargement	56
15. The Conceptual Schema: WARM Data Enlargement	59
16. The Conceptual Schema: Parametric Data	61
17. The O-ODDL Class Specification	66
18. The O-ODDL Inheritance Specification	66
19. The O-ODDL Cover Specification	67
20. The O-ODDL Set_of and Inverse_of Specifications	67

21. The Logical Design: DDL Implementation of the PARAMETRIC DATA Class	68
22. The Logical Design: DDL Implementation of the EMITTER Class	70
23. The Logical Design: DDL Implementation of the KILTING EMITTER and KILTING ADMINISTRATIVE DATA Classes	70
24. The Logical Design: DDL Implementation of the ANTENNA Class	71
25. The Logical Design: DDL Implementation of the SIGNAL Class	74
26. The Logical Design: DDL Implementation of the RECEIVER Class	77
27. The Logical Design: DDL Implementation of the WARM Class	79
28. The Logical Design: DDL Implementation of the SUFFIX TABLE Class	80
29. The Logical Design: DDL Implementation of the S&TI EMITTER and S&TI ADMINISTRATIVE DATA Classes	80

ACKNOWLEDGMENTS

I would like to thank Dr. David Hsiao and Dr. C. Thomas Wu for their guidance in the development of this thesis. And thanks to Tom McKenna and J.J Lee who epitomize the concept of teamwork.

I would also like to thank Doris Mlezko, Sharon Cain, and Donna Colehour for their enthusiastic support.

Most importantly, I would like to thank my wife Maureen and my son Matthew for their endless patience.

I. INTRODUCTION

In this thesis, I propose an object-oriented design for a representative portion of the Electronic Warfare Integrated Reprogramming Database (EWIRDB). In this chapter, I highlight the important role of the EWIRDB in the national defense and provide a description of the current format of the database. I conclude with specific thesis objectives and an outline for the organization of the thesis.

A. AN OVERVIEW OF THE EWIRDB

Advances in electronic warfare (EW) technology have had tremendous impact on modern military operations. The application of electromagnetic energy to secure friendly use of the electromagnetic spectrum, and to detect, reduce, or prevent its hostile use may well be the decisive factor in the outcome of battle. A force that effectively utilizes the electromagnetic spectrum gains the initiative. A force that exploits the weaknesses in an adversary's EW systems renders the adversary blind to the actual tactical situation. Success in EW is a prelude to victory. Failure in EW is militarily devastating.

In the context of today's electronically-dependent warfare, frequent data collection and analysis is essential to the development of EW technologies to counter the enemy threat. Efficient maintenance of the latest data, obtained directly from measurement, or indirectly via electronic intelligence (ELINT), is the basis of successful EW. The National Air Intelligence Center (NAIC) maintains the latest data, in-depth and specific, on EW systems of the United States, friendly forces, and non-friendly forces. These data are stored in the Electronic Warfare Integrated Reprogramming Database (EWIRDB).

"The EWIRDB is the primary Department of Defense (DoD) approved source for technical parametric and performance data on noncommunications emitters and associated systems." [1] Noncommunications emitters include radars, jammers, navigational aids, transponders, target-sensing systems, and others. All such emitters generate and receive electromagnetic radiation and may be used to gain the advantage in armed conflict.

EWIRDB emitter data are therefore indispensable in the analysis and execution of EW; without them, our ability to effectively manipulate the electromagnetic spectrum would be compromised.

The EWIRDB is the union of data from three constituent sources. The National Security Agency (NSA) contributes data from its "Kilting" database. Obtained through ELINT, Kilting data are referred to as observed data in the EWIRDB. Observed data result from the direct measurement and analysis of an emitter's electromagnetic signature following the signal intercept; they are fundamental in describing an emitter's performance. The Scientific and Technical Intelligence (S&TI) community, under the jurisdiction of the Defense Intelligence Agency (DIA), contributes parametric data assessments to the EWIRDB. S&TI systems analysts consider all available sources of information and then estimate or derive the total operational capability of an emitter. Derived parametric data in the EWIRDB are referred to as assessed data. The United States Noncommunications Systems Database (USNCSDB), supported by the Air Force Information Warfare Center (AFIWC), holds data on US owned and operated noncommunications emitters. USNCSDB service analysts provide inputs based on evaluation of system specifications. EWIRDB data of this type take the same format as assessed data, and for this reason, are generally referred to as assessed data as well.

The EWIRDB is thus a data composite. Moreover, this pooling of EW data may reflect different data values from different sources. Figure 1 depicts the EWIRDB as a composite of its three contributory sources.

Developed in the seventies to support the reprogramming of EW systems, the EWIRDB and its role has since grown in both scope and in significance. While its primary focus remains in EW software reprogramming, the EWIRDB has become vital in other areas: EW research, development, test, and evaluation (RTD&E); modeling and simulation (M&S); training and acquisition. Merits of the EWIRDB are revealed by post-

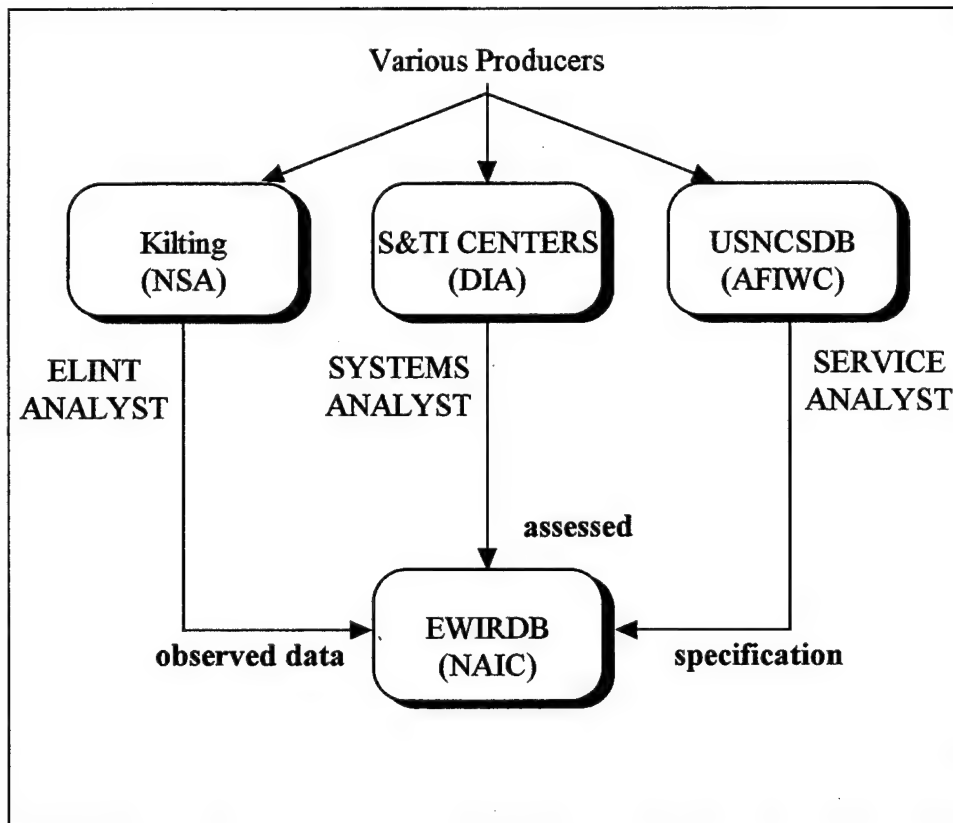


Figure 1. The Merging of Data into the EWIRDB

Desert-Storm figures: the value of the reprogrammable EW equipment directly supported by the EWIRDB has been estimated at \$30 billion; the value of the operational systems, RTD&E, M&S, and training and acquisition programs that employ the EWIRDB has been estimated to be \$1 trillion [1].

In short, the EWIRDB is an indispensable tool that helps to bridge the gap between data analysis and effective exploitation of the electromagnetic environment by EW systems. It is a medium whose use ultimately helps maintain military readiness and minimize the loss of life in combat.

B. THE FORMAT OF THE EWIRDB

Although effective in its implementation, the data model of the EWIRDB is problematic. The EWIRDB is described in terms of a data-implementation model to the exclusion of a legitimate semantic data model. Data is presented in a hierarchical tree that is inherently arbitrary and reliant on the use of reference codes to link related pieces of data throughout the hierarchy. The non-intuitive hierarchical organization and coding scheme prevent the user from gaining a meaningful view of an emitter's performance parameters. Consequently, the nature and semantics of the EW data are obscured by its current representation.

The administrative information maintained for emitter systems and their associated parametric data entries is excluded from the existing data model. The administrative data are addressed only in terms of the formatting of the data output file. This is a major shortcoming; the administrative data are important to the analysis and tracking of parametric data, and represents a significant portion of the database.

In general, the "intuitiveness" of data representations and the ease with which data formats may be interpreted largely determine the usefulness of a database. The current EWIRDB oversteps the boundaries of both criteria. So while it remains the foremost source of mission-critical EW data, lack of an adequate semantic data model ultimately results in a reduction of the EWIRDB's effectiveness as an instrument of EW.

1. The Parametric Tree Model

The upper-level hierarchical data model of the EWIRDB is illustrated in Figures 2 and 3. The Pulsed/Continuous Wave (P/CW) tree in Figure 2 is used principally to evaluate and identify the electromagnetic energy radiated by emitters. The Receiver Parametric Performance (RPA) tree in Figure 3 contains receiver design and performance information on the receiver portion of emitter systems and serves as a vital reference in the development of electronic countermeasures (ECM) techniques and systems. The P/CW and RPA trees together provide a comprehensive report on an emitter's performance. A third hierarchical structure, the Electronic Countermeasures (ECM) tree, exists; it is not shown in any figure. ECM tree data describe jamming systems, and are referenced in the development of electronic counter-countermeasures (ECCM) to overcome the jammer threat. At present, however, the viability of the ECM tree is being reevaluated by the agencies that participate in and contribute to the EWIRDB program. The ECM tree is therefore not addressed in this thesis.

a. The Parametric Tree Structure and Notation

As depicted in Figures 2 and 3, the tree structures graphically show how emitter data are catalogued. "The tree is a management tool that orders a long list logically and hierarchically in a way that proceeds from broad characteristics through levels of successively finer characteristics" [1]. Each branch contains a heading or label to indicate the type of parameters or attributes associated with the branch. For example, "SIGNAL POWER" of the 11 B (B) SIGNAL POWER branch in Figure 2 is a branch name or heading. Branches contain zero or more parameters. A branch with zero subordinate parameters is referred to as a "superheader". Superheader branches pose a unique modeling problem - they contain no data and are not reflected in the data contained within the database. However, superheaders are useful, despite their lack of parametric data, in identifying a major areas of interest to be decomposed in subordinate branches.

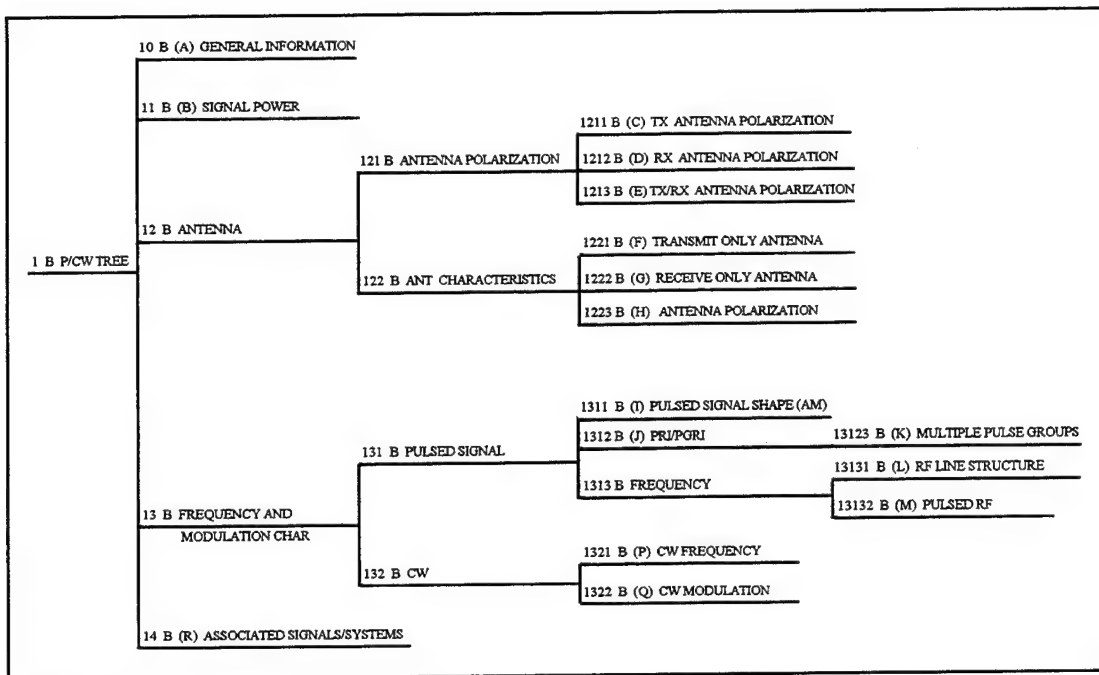


Figure 2. The Pulsed/Continuous Wave (P/CW) Parametric Tree

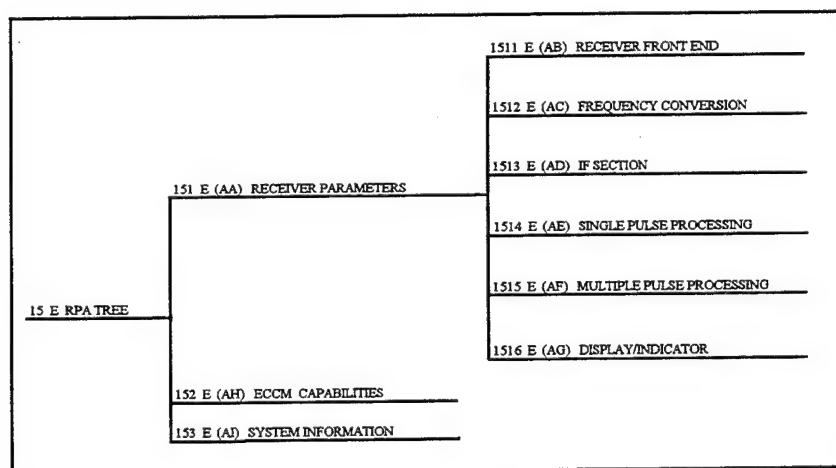


Figure 3. The Receiver Performance Assessment (RPA) Parametric Tree

Thus, in a parametric tree, branches categorize emitter and signal parameters, whereas parameters hold actual data values in the database. A numbering system is also provided for describing branching throughout the depth of the parametric tree. The branch number is given as the first entry on a branch. Each branch has a single predecessor and is assigned a unique number to define a unique path from the root of the tree to any given branch. The "11" of the **11 B (B) SIGNAL POWER** branch in Figure 2 is an example of a branch number.

As specified by branch markers called subfile codes, data are organized throughout the tree to effect logical groupings of parameters. Subfile codes appear in parentheses in Figures 2 and 3. Data subhierarchies rooted at subfile-coded branches are meant to encapsulate major aspects of an emitter's performance or convey the semantics of high-level emitter and signal characteristics. Subfiles are therefore equivalent to subtrees, and accentuate major groupings of related data. The "(B)" listed on the **11 B (B) SIGNAL POWER** branch in Figure 2 indicates that subfile B, rooted at branch 11, contains data that in the composite is descriptive of the high-level characteristic "SIGNAL POWER".

All branches and parameters in the EWIRDB are not applicable to all database users. A branch or subordinate parameter may be useful to an S&TI analyst, for instance, and meaningless to Kilting analyst. Likewise, the data in a particular branch may be applicable to all users. Parametric trees contain usage codes to distinguish usability of branches and parameters among participating agencies. The non-parenthesized "B" on the **11 B (B) SIGNAL POWER** branch, for example, indicates that the SIGNAL POWER branch is used for Kilting, S&TI, USNCSDB, and NSRL (National SIGINT Requirements List) purposes. In other words, that branch is applicable to all agencies that use the EWIRDB. The other codes are K for Kilting and NSRL usage, E for S&TI assessed data and USNCSDB, and N for NSRL-only usage.

The hierarchy depicted in Figure 4 offers perspective on the complexity of the parametric tree. Specifically, all branches subordinate to branch **121 B ANTENNA**

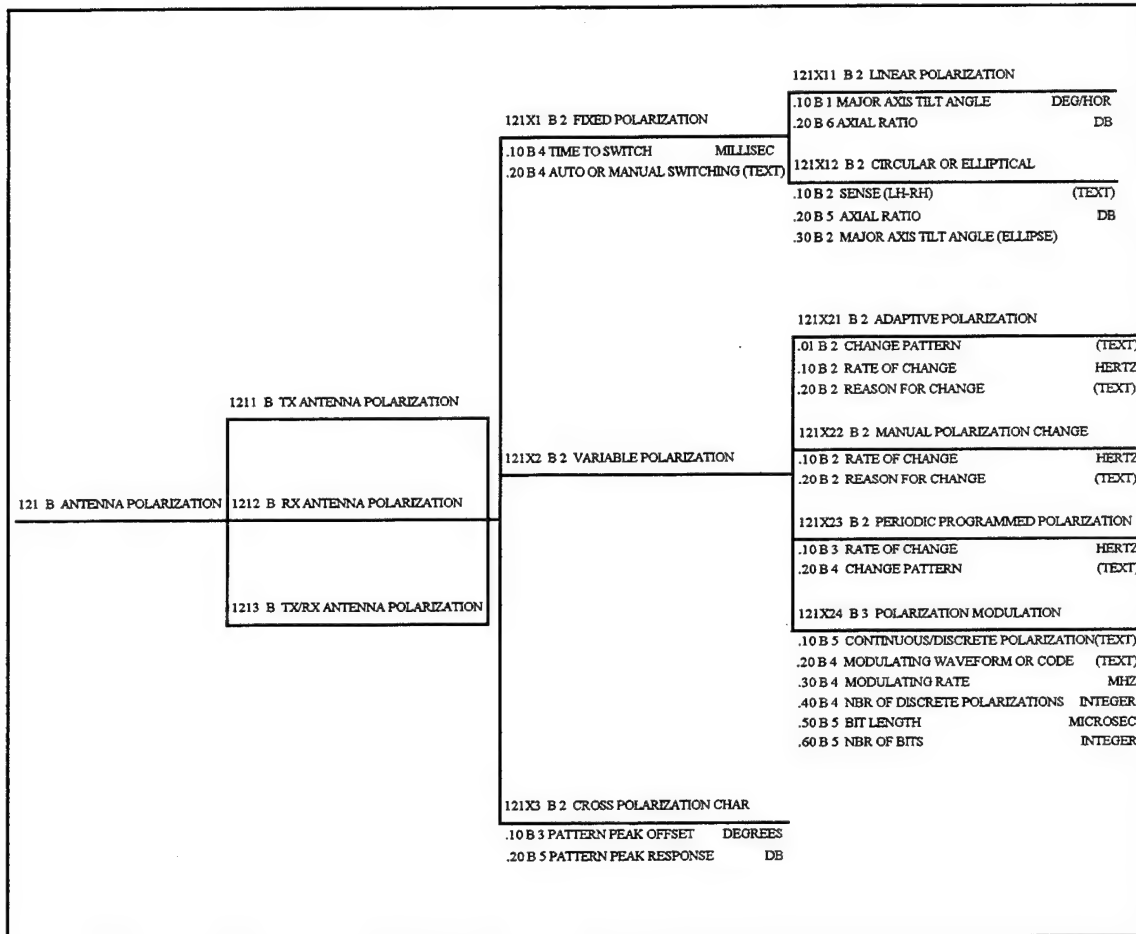


Figure 4. A Detailed Portion of the P/CW Tree

POLARIZATION in the P/CW tree, that is all parameters associated with the branch, are revealed. This portion of the parametric tree is neither the most complex nor the most populated, but it is a precise and representative sampling of the data that reside in the lower levels of the parametric tree.

The new notation in Figure 4 requires a brief explanation. A parameter is listed with a two digit decimal number as a means to differentiate between parameters in a given branch. (Branches themselves include the decimal notation, “.00”, but the notation is implicit and not shown in the tree model.) The combination of the branch number and the two-digit decimal number is referred to as the parametric number. Thus, locating a parameter within the tree or within an output data file is a straightforward function of indexing into the data via the parametric number. For example, parametric number 121X1.10 indexes to the parameter .10 B 4 TIME TO SWITCH under the 121X1 B 2 FIXED POLARIZATION branch in Figure 4. (The X in the branch number is a variable that specifies the type of antenna being considered, i.e., transmit, receive, or transmit and receive. The variable takes on the value 1, 2, or 3, accordingly.)

Additionally, since each parameter contains data, each includes an entry for units of measure. Branches, in contrast, are not data entries but rather indicate that parametric data groupings may be identified by a branch name or number, and therefore do not specify units of measure.

b. The Limitations of Hierarchical Data Modeling

In general, the hierarchical data modeling of the EWIRDB parametric trees is misleading in its representation of parametric data. Aside from highlighting the complexity of the EWIRDB parametric tree, the sample hierarchy in Figure 4 also exposes the arbitrary nature of the trees' hierarchical structure. An inability to precisely represent data semantics is common to generic tree structures such as those of the EWIRDB. The current EWIRDB tree model is strapped with this inherent arbitrary quality that limits the

EWIRDB's effectiveness as a database and places the burden of data interpretation on the user.

Specifically, the parallel branches, **121X1 B 2 FIXED POLARIZATION**, **121X2 B 2 VARIABLE POLARIZATION**, and **121X3 B 2 CROSS POLARIZATION CHAR**, seem to indicate that for a given antenna, polarization is either fixed or variable or exhibits cross polarization characteristics. This is not actually the case. For a given antenna, polarization is either fixed or variable, and all antennas may be described by cross polarization characteristics. Whereas the fixed and variable polarization branches determine a clear boundary based on fundamental differences in an antenna's characteristics, the cross polarization branch is applicable to all antennas, regardless of their differences. The hierarchical structure in Figure 4 does not convey this idea. It provides only a generic and inadequate treatment of the intended data semantics.

A similar situation arises in the hierarchy rooted at branch **121X2 B 2 VARIABLE POLARIZATION** in Figure 4. The arbitrary nature of the hierarchical modeling structure depicts a variably polarized antenna that appears to be rigged as one of four types: adaptive, manually changed, periodic programmed, or modulated. Again, this does not accurately reflect the intended meaning of the data. The correct interpretation is that a given variably polarized antenna can be described as one of three types: adaptive, manually changed, or periodic programmed. And just as the cross polarization branch applied to any given entry in the preceding antenna polarization branch, the polarization modulation branch describes characteristics common to all variably polarized antennas. The polarization modulation is therefore not a criteria by which to categorize types of variable polarization.

Another flaw in the EWIRDB tree model is a collateral effect of the general layout of the data. Parametric data is scattered over a large number of separate records comprising two distinct and largely independent structures, the P/CW and RPA trees. A search of these two distinct structures and their associated parameters is required to

ascertain the performance of a given emitter. Consequently, the global view of an emitter's performance, from a modeling perspective, is obscured.

Deficiencies in the parametric tree model are further exacerbated by the fact that the trees are designed to characterize only parametric data. The EWIRDB also contains administrative, reference, and commentary information, all associated with parametric data. At best, then, even if the trees were perfect parametric data modeling tools, only a portion of EWIR data would have been taken into account.

The data not included in the parametric tree are loosely modeled in terms of a file structure. The file structure is not, however, a data model. It is a description of the data as presented in the output form. Parametric data is therefore also described in terms of the output format. While the file "model" incorporates all aspects of the database, the overall semantic picture is difficult to grasp; the file format is also complex and disjoint.

2. The File Structure of the Output Data

The EWIRDB output file format is designed to provide a comprehensive view of parametric and associated data for emitters. It is cryptic in presentation, however, and does not compensate for the lack of a semantically correct data model. While the view of an emitter's parametric and associated data in the output file is complete, it is non-intuitive. The Technical ELINT Reference File format (TERF) is the standard distribution format for the EWIRDB and is composed of six different types of records, referred to as logical information records. The record types are specified as follows, with the record name preceding the record type designator in parentheses: **Classification Record (S00)**, **Emitter Name Record (S01)**, **Subfile Header (S02)**, **Parametric Data (S03)**, **Reference Data (S04)**, and **Comments (S05)** [1].

A brief description of the TERF data fields is required to bridge the gap between the data as modeled in the parametric trees and the data as presented in the TERF output. Because the EWIRDB consists of data merged from different sources (see Figure 1), some

fields are source-specific. A tabular summary of the parametric data and other types of data in the output file is provided in Figure 5. In the figure, “assessed data only” refers to both S&TI and USNCSDb contributed data, as stated earlier in Section I.A. A full description of the TERF format, including the actual “look” of an output file, is given in [1].

Three fields do not appear in Figure 5 but are common to all records in a file. The first is *Record Type*, which specifies the record as S00, S01, S02, S03, S04, or S05. The second field is the *Source Designator*, which identifies the contributory source of the data contained in that record; K for Kilting, E for S&TI assessed data, and U for USNCSDb. The third field is *Notation*, which provides the ELNOT (ELINT Notation) assigned to the given emitter. The ELNOT is an administrative label that uniquely identifies an emitter.

Overall, the TERF format is complex. It represents a merger of data from different sources with different needs and provides for nonstandard, source-specific data formats. The TERF contains many codes. Some codes differ in symbology but relate to identical components, and some apply to only certain types of data. Other codes distinguish between multiple versions of the same parameter, and some relate mutually dependent parameter values. Mode combinations and the suffix table pose a particularly challenging modeling problem. While modal relationships are critical in the identification and evaluation of emitters, the relationships as coded in the suffix table are difficult to grasp, especially if emitter modes number in the hundreds of thousands. (Suffix codes are given more detailed treatment in Chapter IV).

Many TERF fields exist solely to link information in one portion of the file to information in another segment of the file. The coding and linking picture grows more complex within the following context. A TERF consists of emitter data partitioned into subfiles represented in the S02 records. Each contributory source (Kilting, S&TI Assessed, USNCSDb) may supply many different subfiles for a given emitter, each may supply multiple versions of the same subfile, and sources may overlap in the subfiles they

RECORD	RECORD NAME or TERF FIELD	DESCRIPTION
S00	Classification	one S00 per emitter
	<i>Classification</i>	overall classification of emitter file
	<i>Retrieval Date</i>	Kilting only, data of data extraction from NSA database
S01	Emitter Name	one S01 per contributory source (K,E,U)
	<i>Emitter Name</i>	name commonly associated with the ELNOT
	<i>S&TI Code</i>	assessed data only; 4 character code that identifies the agency responsible for the ELNOT
	<i>SAE Code</i>	Kilting only; 4 character code that identifies the agency responsible for the ELNOT
	<i>Multiple Source Review Date</i>	assessed data only, date of the last full review of the assessed data file for a given emitter
	<i>Date of Last Significant Change</i>	Kilting only, date of last full review of the Kilting data file for a given emitter
	<i>Parametric Update Date</i>	date of most recent change to any S03, S04, or S05 record
S02	Subfile Header	one S02 per parametric data subfile per contributory source; multiple S02 records likely
	<i>Subfile Tree Number</i>	subfile-coded branch number
	<i>Subfile Name</i>	name (heading) of the subfile-coded branch
	<i>Subfile Code</i>	1 or 2 character code denoting the subfile or subtree
	<i>Technical Date</i>	Kilting only, date of last change in any S03 record
S03	Parametric Data	one to many S03 records per parametric data entry per contributory source; multiple S03 records likely
	<i>Tree Number</i>	also called parametric number; index into parametric tree
	<i>Suffix Code</i>	1 or 2 character code assigned to help describe emitter modes; helps differentiate between multiple entries for the same tree number; links related (dependent) parameters
	<i>Measurement Name</i>	corresponds to branch/parameter name in parametric tree
	<i>Units</i>	corresponds to units specified for parameters in parametric tree; for textual data, the format may be specified here
	<i>Lower/Upper Value or Text</i>	actual parametric data; for numeric data, lower/upper value is filled in (with same values if data is single-valued)

Figure 5. A Description of TERF Elements (continued into next page)

RECORD	RECORD NAME or TERF FIELD	DESCRIPTION
S03 cont'd	<i>Confidence Level</i>	assessed data only; specifies the analyst's confidence in the parametric data
	<i>S&TI Code</i>	assessed data only; 3 character code that identifies the agency responsible for the ELNOT
	<i>Reference Number</i>	links S03 to a reference (S04); 4 character code that refers to a line in an S04; 1 st character in code denotes the data source, R=Kilting, A=S&TI Assessed, F=USNCSDb (differs from S01 code)
	<i>Comment Number</i>	code that refers to a line in an S05; 1 st character in code denotes the data source; C=Kilting, K=S&TI Assessed, N=USNCSDb (differs from S01 and Reference Number codes)
	<i>Reserve Mode</i>	code to indicate that the value of the parametric data, or mode, is a wartime reserve mode (WARM); also indicates analyst's confidence in this assessment
	<i>Classification</i>	assessed data only; U=unclassified, C=confidential, S=secret, or T=top secret
	<i>Releasability</i>	assessed data only; 2 character code designating the countries to which the data is releasable
	<i>Date of Last Update</i>	date the last significant change was made to the data
	<i>Measurement Accuracy</i>	Kilting only; + or - range if available, used with numerical parametric data
	<i>Measurement Accuracy Units</i>	Kilting only; same as the units field unless the accuracy is so fine it cannot be expressed the same way
	<i>Intelligence Source</i>	Kilting only; 1 character code, denotes type of source used to derive parametric data (ex. ELINT, non-ELINT)
	<i>Preferential Rating</i>	Kilting only; one digit code to signify the relative importance of the data, the importance of obtaining the data
S04	Reference Data	zero to many S04 records per source per emitter file; required if a reference was specified in an S03 record; provides a trace back to original source documents
	<i>Reference Number</i>	same as those specified in the S03 records
	<i>Reference Line Number</i>	sequential and contiguous; many lines of text may be required to describe a reference for a given reference number

Figure 5 cont'd. A Description of TERF Elements (cont'd into next page)

RECORD	RECORD NAME or <i>TERF FIELD</i>	DESCRIPTION
S04 cont'd	<i>Reference Text</i>	<ul style="list-style-type: none"> assessed data format: textual description of the parametric data reference followed by a formatted classification/releasability line (refers to the S04) Kilting format: reference text or document number (document title), report date, producer, classification of the report
S05	Comments	zero to many S05 records per parametric data item per source; required if a comment was specified in an S03; suffix table stored in "comment zero"; general emitter comments stored in "comment one"
	<i>Comment Number</i>	same as those in S03 records
	<i>Comment Line Number</i>	sequential and contiguous; many lines of text may be required to describe a comment for a given comment number
	<i>Comment Text</i>	used to explain, describe, elaborate, and qualify parametric data entries and modes <ul style="list-style-type: none"> assessed data format: includes a formatted classification line for every comment; at least one classification line is required for each comment

Figure 5 cont'd. A Description of TERF Elements

supply to the EWIRDB. Each subfile in turn may consist of many different parametric data entries, and there may be many data entries for the same parameter as represented in the S03 records. Finally, where applicable, parametric data links to source-specific reference documentation and comments in the S04 and S05 records, respectively. And for a given emitter, each source may require many S04 and S05 records. The effect of the data merge, codes, and links with this framework is an elaborate and burdensome presentation of parametric and associated data.

3. Summary

The EWIRDB represents a challenging database modeling problem. The problem stems from several factors, the foremost of which is the inherent complexity of the data. Capturing the nature of EW systems and signals is difficult.

Additionally, the parametric trees, the semantic basis of the EWIRDB, have been designed and used primarily for database management, not as data modeling tools. To the extent that the trees have been used to model parametric data, their hierarchical and intrinsically arbitrary structure has proven too restrictive to accurately capture the semantics of the data. The database user is therefore required to logically determine the true nature of the data, if the need for interpretation is recognized at all.

Further, TERF-formatted EWIRDB output provides a comprehensive view of emitter data, but does not fill the semantic gap. While it incorporates the structure of the parametric tree model and catalogues associated reference and commentary data, it cannot be construed as a data model. Moreover, the TERF format introduces extras into the data, such as reference codes, to link related pieces of information. The use of codes throughout the file muddles the meaning of the data.

Finally, without system-supported semantics, the burden of EWIR data interpretation is transferred to the user. This is not an easy task for the user; the EWIRDB is difficult to comprehend because the nature and relationships of EW data are not adequately modeled and are subject to coding. Because the EWIRDB is generally

described in terms of data implementation and not data semantics, there exists a requirement for the development of a more meaningful, intuitive, and system-supported design. The recent advance in object-oriented data modeling indicates that the object-oriented alternative may prove useful in simplifying and clarifying the data semantics, relationships, and formats of the EWIRDB.

C. THESIS OBJECTIVES

The primary objective of this thesis is to provide a new object-oriented design for a sample portion of the EWIRDB. NAIC has identified the EWIRDB for our experimentation in object-oriented database design. The object-oriented data model is arguably the most semantically rich and flexible of all database design tools. The effectiveness of the object-oriented data model, however, remains untested for any military or warfare-related design of the scope of the EWIRDB.

The secondary objective is to use the object-oriented data definition language (O-ODDL) as a design tool for the specification of the object-oriented EWIRDB. At present, the O-ODDL used in this thesis is the product of a larger thesis effort that produced an object-oriented interface to the Multimodel and Multilingual Database System (M²DBS) [7] at the Naval Postgraduate School (NPS). The O-ODDL specification of a new EWIRDB design is therefore a continuation of the NPS research. It will ultimately provide an on-line object-oriented EWIRDB with which to demonstrate both the utility of the new M²DBS object-oriented interface, and the usefulness of the new object-oriented EWIRDB design.

D. THE ORGANIZATION OF THE THESIS

In Chapter II of the thesis, I address basic issues in the object-oriented database development, within the context of conceptual design and logical design processes. In Chapter III, I provide the design mechanisms of the object-oriented data model. In

Chapter IV, I further describe the tools of the proposed object-oriented design and present the conceptual design of the EWIRDB. In Chapter V, I briefly describe the logical design structures native to the M²DBS and present the logical design. In Chapter VI, I summarize my assessment of the new object-oriented EWIRDB.

II. DESIGN CONCEPTS

Database design is a multiphase process. Each phase addresses a different aspect of the design process and yields a separate design result or model. The partitioning of the design process in this way guarantees the viability of each design phase as a distinct entity. Moreover, it simplifies the entire process, because the complexity of the design problem is also partitioned. Only certain aspects of the design need be addressed in each phase, and the designer is exposed to the details of a given level only. The correct and thorough design of one phase lends itself to the development of a subsequent phase.

In this chapter I addresses those aspects of database design that are central to this thesis: the *conceptual design* and the *logical design*. These are the first phases in the overall design process and are therefore elemental to the overall design. Together, the conceptual and logical design phases take a proposed database from abstraction to implementable form.

The treatment here is generic; design mechanisms specific to object-oriented database design are examined in Chapter III.

A. THE CONCEPTUAL DESIGN

Much like an architect's sketch crystallizes the customer's architectural design vision, the conceptual design captures the nature of data in a way that closely resembles the database users' perception of data and the usage of data.

The fundamental goal of conceptual database design is thorough understanding of the database through development of a *conceptual schema*. A tool known as the *high-level data model*, also referred to as a *semantic* or *conceptual data model*, is used. A high-level data model is intuitive, flexible, and comprehensive in its description of data. It is the means by which a schema is developed to approximate the users' perception and usage of the proposed database. To this end, the set of abstraction concepts underlying

the semantic data model are sufficiently expressive of data, simple in nature, unambiguous, minimal in number, and nonoverlapping in meaning [2].

Devised within the framework of the high-level data model, the conceptual schema thus characterizes the structure of data. The structure of the data is the sum and substance of the database, encompassing data types, data relationships, and data constraints. Since the conceptual design should be intuitive, its design notation is typically associated with a diagrammatic representation of its modeling constructs. A diagram is a simple, precise, high-level, and straightforward means of expressing the nature of data.

An essential quality of a conceptual schema is that it be independent of a specific database management system (DBMS). A DBMS-independent semantic data model is generic and free of any limitation or peculiarity imposed by a particular DBMS. Consequently, the details of data implementation and physical data storage are suppressed in the conceptual schema. Such detail is not useful in the development of a high-level conceptual design. Accordingly, the conceptual schema cannot be used directly to implement the database. This, however, is not disadvantageous. Rather, it highlights the importance of the conceptual design and the value of the conceptual schema as a stable description of the database. A stable database description - the conceptual schema - remains unaltered by any modification to the underlying DBMS-dependent logical and physical designs.

As the initial phase in the design effort, conceptual design is paramount in database development; the entire process depends on the creation of a stable and correct conceptual schema.

B. THE LOGICAL DESIGN

The architect's initial sketch, like the conceptual schema, is the foundation for all subsequent design work. After capturing the essence of the customer's desires in the sketch, the architect then addresses the specifics of the design layout. Decisions are made

based on the environment and the available materials. The outcome is a blueprint, a specification for the construction of the design.

As the blueprint follows the sketch, the logical design in database development follows the conceptual design. The logical design likewise yields a "blueprint" of the conceptual schema that accounts for the type of database system in which the database will reside.

The logical design is equivalent to a mapping from conceptual schema to the data model of the selected DBMS. The mapping is accomplished by the designer via the DBMS's native data definition language (DDL); the output DDL statements are equivalent to a DBMS-readable specification of the conceptual schema. The end result of logical design is thus a transformation of the database as proposed in the conceptual design to a database in the DBMS-compatible form for eventual realization in the DBMS.

III. OBJECT-ORIENTED DESIGN CONCEPTS

Conceptual design and logical design, as described in Chapter II, cast the foundation of database development; a high-level data model provides the mechanisms required to formulate these designs. Thus, both design processes proceed within the framework of the chosen data model. The data model is therefore the starting point.

The definitive measure of a data model's effectiveness is its abstraction capability, or the degree to which its design mechanisms capture "real-world" semantics. Traditional data models, including the hierarchical model, are limited with respect to their abstraction capabilities. The EWIRDB hierarchical model is a prime example; and as detailed in Chapter I, the model is fundamentally deficient in its representation of EWIR data. For traditional data models in general, the more complex the nature of the data, the greater the semantic mismatch between the real-world data and its representation.

Object-oriented database design, a departure from traditional methods, seeks to eliminate the semantic mismatch between real-world entities and their database representations. The *object-oriented data model* (OODM) is the basis of the design effort. The OODM is more semantically rich than the earlier models. Object-orientation more closely parallels the way we observe the real-world. We are surrounded by objects: computers, cars, roads, buildings, trees, people, animals, the atmosphere - the list of objects is infinite. People tend to reason about real-world "objects" in terms of their characteristics, both static and dynamic. A car, for instance, might be classified by its make, model, and year, as well as by its performance in various driving conditions. We also tend to apply different degrees of abstraction to the real-world entities that we encounter. Depending on a person's point of view, a real-world "object" may be looked upon as a single, indivisible unit, or as the composite of a number of component objects. Returning to the car example, the typical car owner probably takes the view that a car is an integral unit that provides a means of transportation. A car mechanic, on the other hand, probably sees a car as the sum of its parts - parts that require maintenance and replacement. The object-oriented approach is a close approximation to these human views

of the world. It is for this reason that object-oriented abstraction techniques are generally considered to be more powerful than those of the traditional data models.

The OODM thus provides the design mechanisms with which to model diverse and sophisticated applications in a natural way. In a larger sense, within the context of overall database development, the object-oriented approach reflects a move toward an “intelligent” DBMS that directly supports advanced data modeling. In such a system, semantic correctness remains intact from abstraction to implementation. The burden of translation is lifted from the user.

The object-oriented paradigm remains the focus of the active research. While researchers and developers agree on the underlying principles, the exact nature and direction of the object-oriented approach is at present an issue of debate. Consequently, a final and irrefutable definition for the OODM has not yet been forwarded. Despite the evolutionary condition of the OODM, the motivation to preserve a direct correspondence between real-world entities and their database representations warrants its use. The EWIRDB is an ideal candidate for object-oriented modeling.

In this chapter, I present the basic concepts of the OODM. Because the OODM was developed with the ease of implementation in mind, some implementation issues are also briefly addressed. These concepts lay the groundwork for an application of the OODM, within the context of both conceptual and logical design, to a representative portion of EWIRDB data in Chapter IV.

A. OBJECTS

The *object* is the basic element of the OODM, and the component that populates the database. An object corresponds to any entity in the real world: ideas, concepts, people, events, places, physical structures, and time to name a few. The uniform application of objects to model the spectrum of real-world entities simplifies the designer’s view of the real world [4] and infuses some consistency into the designer’s task.

In an object-oriented database management system (OODBMS), an object is specified with a unique, system-generated marker called the *object identifier (OID)*. The OID is immutable, or permanent and unchangeable [2]. This is an important aspect of the OODM from a modeling and implementation point of view. The use of OID's effectively decouples the object existence from the object value. An objects can therefore be referenced via the OID, independent of an identifying value. Two objects with different OID's remain distinct, even if the two objects have the same values. In traditional models, on the other hand, the identities of data items are value-based. The cumbersome task of creating and managing unique identifiers (called keys traditionally) is therefore imposed on the application programmer. Consequently, meaningful keys are likely long and non-unique, and the management of key values is carried out external to the DBMS. The effect is a degradation in database performance.

The hierarchical model of the EWIRDB is value-based and therefore subject to these shortcomings. Specifically, data items referenced by application programs steer through an identification scheme that includes the ELNOT and a burdensome hierarchical labeling network. For a given ELNOT, or equivalently for a given emitter, a data record is uniquely identified by a suffix code/tree number/source combination [1]. In an object-oriented EWIRDB, a data object is uniquely identified by a system-maintained OID.

The OODM also provides for the creation of objects of arbitrary complexity [2]. The internal structure of objects is thus sufficiently adaptable to include all significant information that describes an entity. This internal structure is referred to as the object's *state* and *behavior* [3]. These aspects of the OODM are addressed in the following sections.

1. Object State

An object is characterized by internal properties generally referred to as *attributes*. The values of an object's attributes define the *state* of the object. Attribute values may either be simple or complex.

a. Simple Attributes

Simple attributes are those whose values are *literals* - character strings, integers, floating-point numbers, and other primitive values. Typically, literals are not considered as objects. For efficiency reasons, they are usually represented directly or are self-identifying, and not associated with OID's [4].

b. Complex Attributes and Relationships

Complex attributes are those whose values are composed of other objects or groupings of values. There are three kinds of complex attributes: *reference attributes*, *collection attributes*, and *derived attributes* [4]. The first two types provide for an arbitrarily deep or recursive nesting of objects, where the state of an object is described by attributes whose values are objects whose values may be objects as well, and so on. A natural representation, then, for the state of an object is a set of OID's of the objects that are the values of the attributes of the object [3].

Reference attributes are the means by which relationships between entities are represented in the OODM. In taking on object values, reference attributes explicitly refer to, or draw a relationship to, other entities. Specifically, in the logical design, reference attributes may be used to model binary and non-binary one-to-one, one-to-many, and many-to-many relationships. A relationship may be modeled in one direction, such as from an object A to an object B, where object A refers to object B but object B contains no such reference to object A, or in both directions through the use the of an *inverse reference or inverse attribute* [2]. An inverse reference facilitates traversal of the relationship. The relationship is "visible" to each object; object A refers to object B *and* object B refers to object A inversely. All the relationships in which a particular object type participates are thus packaged within the object itself in the form of reference attributes. In contrast, a complete inspection of the parametric trees and TERF output may be required to ascertain the relationships that exist between particular parametric entities in the EWIRDB.

In implementation, reference attributes provide an additional benefit. They cannot be corrupted, i.e. inadvertently or maliciously changed: the integrity of relationships and references is maintained by the OODBMS throughout all database operations. Moreover, from a modeling perspective, because a reference attribute refers to an OID and not a value, the values encapsulated within the object to which the reference attribute points may be changed with no effect on the OID, and thus no effect to the reference attribute.[4] The use of reference attribute has one possible shortcoming, however. Beyond meaningful reference attribute names, references in the OODM do not imply any special semantics. Basically, references can only convey the idea of an association between entities.

Collection attributes encompass those characteristics of an object that are described by more than one value, or present a complex arrangement of values. These values are stored in constructors such as lists, sets, or arrays. The value sets, or domains, from which the values comprising the collection are taken may contain simple values or references. For example, a collection attribute may be a set of integers or a list of entities that participate in a relationship with the object.

Object properties that are subject to frequent or regular modifications, such as those that are time-based or date-based, are best modeled with *derived attributes*. Derived attributes, as the name implies, are not stored explicitly. Rather, they are defined via the execution of a particular procedure. A given value for a derived attribute, and therefore its storage, is temporary in nature.

Except for the brief introduction to derived attributes, the discussion of object state to this point has dealt with the static characteristics, or *structure* of an object. The next section addresses object characteristics that are dynamic in nature.

2. Object Behavior and Encapsulation

An important aspect of the OODM is its ability to incorporate the operations to be applied to an object of a certain type into the object itself. The procedures that modify or

return the state of an object in an OODBMS are called methods. The *behavior* of an object is thus defined by the methods specified to act on it.

Methods are much like programs. They are written in a typical programming language. A method consists of two parts: an *external interface* (or *signature*) and the actual code to implement the method. The external interface defines the parameters whereby an object interaction is recognized. It is the only legal means by which to invoke the method. Typically, the execution of a method is accomplished via the *message passing* [2]. If, for example, an object A sends a properly-parameterized message to an object B in order to invoke a method in object B that returns the data stored in object B, then the method of object B would return the data to requesting object A. This concept of restricting access or providing well-defined access to an object is referred to as *encapsulation*. If strict encapsulation is enforced, then the object itself - its internal structure and methods - is accessible only via the specified parameters. The only "user-visible" portion of the object is the external interface; the data contained within the object and the details of the method's implementation are completely hidden from external users. Procedures that are visible outside the object are *public methods*. An object may also encapsulate *private methods*, or those available only to the object itself. In practice, however, strict object encapsulation is too restrictive in any OODBMS [4]. In addition to the public methods, attributes may be made visible as well.

Encapsulation is a basic tenet in the OODM. Its benefit is straightforward: encapsulation permits a change in the implementation of objects without forcing any change in the external programs that use them. As long as external interfaces remain the same, the means to access and manipulate objects remain the same. Provided the external interface remains intact, it follows that objects whose structure has been modified will appear unchanged to the external world. Encapsulation is also important in introducing the concept of *object class*.

B. OBJECT CLASSES

A database generally contains clusters of similar objects. Each cluster contains objects that encapsulate the same structure and behavior, or attributes and methods. Just as an abstract data type is the specification for a number of data structures in a typical program, a *class* in the OODM is the specification for a number of similar objects in a database. A database containing multiple clusters of similar objects would therefore be comprised of several classes. And just as identically-formatted data structures may contain different stored values, objects of similar structure and behavior, or objects in the same class, may exhibit different states.

These ideas are illustrated in Figure 6 which represents a small portion of data maintained in a fictitious database at NPS. The **THESIS** class definition provides the blueprint for creation of **THESIS** objects. This definition specifies three simple attributes - **title**, **author**, and **date of publication** - and two methods - **author bio** and **number distributed**. The method **author bio** returns the author's branch of service and warfare specialty (data stored elsewhere in the fictitious database). The method **number distributed** returns for a given thesis the number of copies distributed, a value that may be subject to periodic change. The class **THESIS** is void of any actual data, but the objects of class **THESIS** contain values for each specified attribute and invoked method. These attribute and method values differ from object to object; each object of class **THESIS** therefore exhibits a different state.

Classes are the basic building blocks of the object-oriented modeling. The concept of class is therefore the basis of fundamental modeling mechanisms in the OODM. These modeling mechanisms are the focus of this section. Some of these mechanisms are considered to be core concepts in the model. The semantically-important *is-instance-of* relationship is one. The concept of *generalization-and- specialization* is another. Less-widely-acknowledged object-oriented modeling concepts of *aggregation* and *covering* are addressed as well.

CLASS DEFINITION

THESIS	
attributes:	
	title
	author
	date of publication
methods:	
	author bio
	number distributed

INSTANTIATION

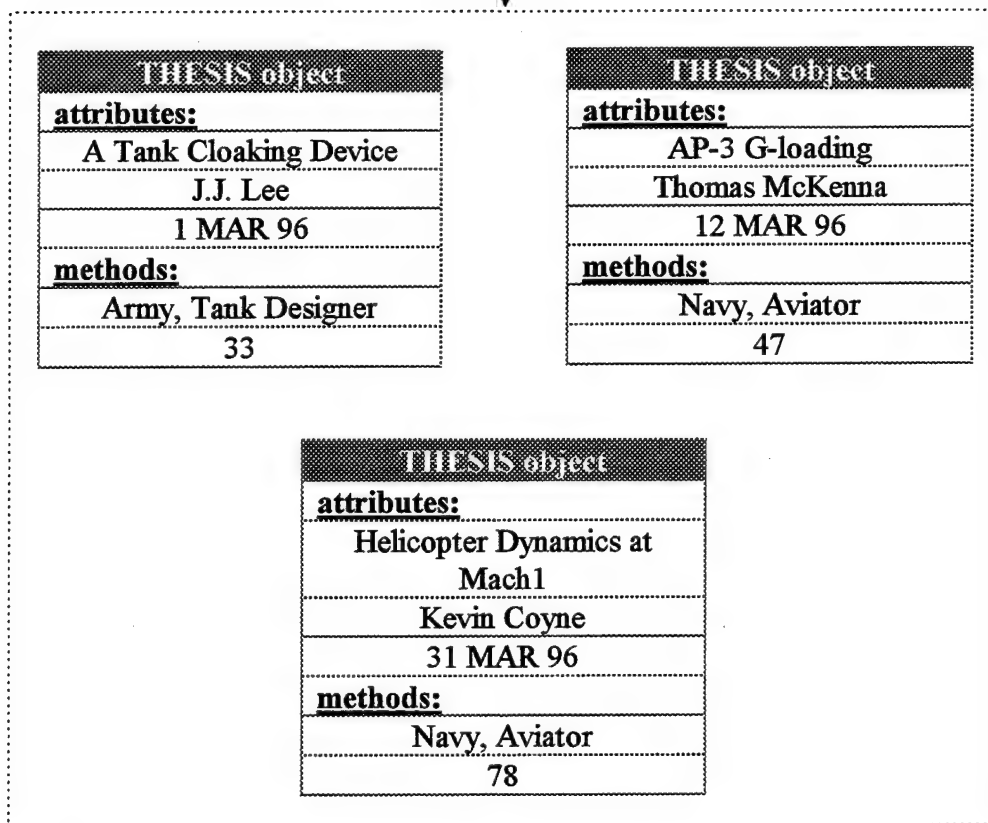


Figure 6. An Object Class and its Objects

In addition to its value in the data modeling, the class concept has important and favorable consequences in implementation. When viewed as the collections of their instances rather than as the specifications of individual objects, classes form the logical basis for the formulation of queries [5]. Further, because attribute and method specifications common to objects of the same class are stored as a *class object*, there is no need to replicate the common information in each object of the class. The effect has considerable savings in storage space. Finally, the class concept provides a degree of "type checking" throughout a *class composition* hierarchy [3]. The class composition hierarchy is the direct result of the recursive nesting of objects as attribute values, an idea introduced in section III.A.1.b. These objects are restricted in their values by their respective class specifications. In this sense, the class is analogous to the traditional notion of attribute domain. Just as the domain defines legal values or types for a given attribute, the class defines the legal values for a particular object of that class. The class thus provides a degree of type checking for an attribute whose value is an object.

With the OODM concepts of the object and the class as building blocks, the following sections detail the design abstractions applied to the proposed object-oriented design of EWIRDB data.

1. Instantiation and Classification

The class itself is an object, void of actual data. Thus, it is also termed the class schema. It functions as the "blueprint" with which to generate objects of the same class. Viewed in this light, an object based on the blueprint of a given class can be thought of as an *instance* or an *occurrence* of that class. Since a class contains the definition of a set of objects, it is also an abstraction mechanism [5]. The class abstraction is rooted in the complementary semantic modeling concepts of *instantiation* and the *classification*.

The instantiation is the process of creating objects within the parameters of a given class schema. Classification is the inverse of instantiation. It is a process of systematically

assigning objects of similar structure and behavior to their respective object classes. Classification permits the modeling of common characteristics that apply to all of the objects in the class.

Because its a single blueprint from which many objects may be created and catalogued, the class structure may be reused as required to instantiate many similar objects. In Figure 6, the blueprint for the class **THESIS** is used three times to instantiate each of the three **THESIS** objects shown. For this reason, instantiation and classification are collectively considered to be the first *reusability mechanism* of object-oriented design. *Inheritance*, addressed in the next section, is the second such mechanism.

2. Generalization and Specialization

Inheritance among classes produces class hierarchies that characterize the OODM abstraction concepts of the *generalization* and the *specialization*. In an inheritance hierarchy, a class referred to as the *subclass* inherits the structure and behavior of another class called the *superclass*. In addition to its inherited characteristics, the subclass may encapsulate attributes and/or methods not contained in the superclass. These distinct additions to the subclass differentiate it from the superclass and identify the subclass as worthy of a class status all its own. In the hierarchy, a subclass is viewed as a specialization of its superclass. Likewise, a superclass can be perceived as the generalization of those subclasses (from one to many) participating in the inheritance hierarchy. Collectively, the concepts of generalization and specialization are equivalent to the is-a-kind-of relationship. If an independent and unique subclass X1 inherits the attributes and methods of a superclass X, then X1 may be considered "a kind of" X.

A data hierarchy based on the inheritance is natural and well-defined, unlike hierarchies based on arbitrary and coded tree structures, such as those found in the EWIRDB. Inheritance emphasizes both the commonality and the uniqueness among classes. Moreover, the implementation of an inheritance (i.e., a generalization and a specialization) as a mapping from class to another class eliminates data duplication and

localizes the management of common data. It is for this last reason that inheritance is touted the second reusability mechanism of object-oriented design.

3. Aggregation

The *aggregation* abstraction considers a composite object as the sum of its parts. It is not restricted to an object as an aggregation of its attributes. The term is primarily meant to represent an object as an aggregation of other objects, i.e., a composite object as the sum of its component objects. The semantics are comparable to those of the *is-a-part-of* relationship, where an entity is the grouping of its components.

The objects of component classes participating in the aggregation each have their own state. Likewise, each object of the composite class exhibits its own state. But the state of the composite object in a given aggregation is dependent upon the states of its component objects. A composite object thus contains a "global" type of structure and behavior that reflects the composite state of its component parts.

Simply drawing a relationship between an object and its aggregates is not semantically sufficient; it does not capture the dependency between the composite object and its components. From an implementation point of view, a relationship will not maintain the integrity of the aggregation, or the interactions within the aggregation, throughout all possible database operations. In particular, an operation on the composite object should affect component objects. Conversely, an operation on a component object should affect the composite object. The deletion of a composite object, for example, should cause deletion of all components of the object. The aggregation and the notion of a composite object can also be used as the basis for the clustering of data [4].

The aggregation abstraction is an important semantic concept in the OODM. It is a design concept not found in other models.

4. Covering

The *covering* abstraction is accepted as a fundamental concept in the OODM within the European community. It adds a dimension of flexibility in the modeling and manipulation of data. The covering terminology is as follows: class X *covers* class Y if every object in class X *corresponds* to a subset of objects in class Y. These subsets of Y need not partition Y; they are certain subsets of all the subsets generated for the objects of Y. Mathematically, all the subsets of Y form the *power set* of Y, i.e., $P(Y)$. The correspondence is a mapping f which determines for an object, x , from class X all the objects, y 's, of the subset $f(x)$ from class Y, such that $f(x) = y$ for every one of those y 's. Class X is referred to as the *cover class* and class Y is called the *member class*. [6]

A covering relationship thus corresponds an object of one class to a subset of the power set (the set of all subsets) of objects of another object class. It is therefore an object-to-object-set mapping.

A simple and practical example involving a team and its players is useful in describing the covering relationship between two classes. In this example, the team class covers the player class. The team's existence is entirely dependent on the participation of its players, a type of existence dependency. While a team object has its own structure and behavior, its real value is derived from its encapsulation of the nature of a particular set of players that comprise the team. Further, a team object may be operated on as a single object or as a set of player objects. And as is generally the case in the real-world, the elimination of a team (object) does not necessarily entail the demise of its players.

The covering is a valuable abstraction mechanism, specific to the OODM, that accurately models entities of the real world.

IV. A CONCEPTUAL OBJECT-ORIENTED EWIRDB

In this chapter, I apply the principles of the OODM, as presented in Chapter III, to develop a genuine conceptual design for the EWIRDB. My intent is not to redefine the kinds of data required to characterize an emitter's performance; the existing EWIRDB data items have sound scientific roots. Nor do I attempt to address every existing data element in the EWIRDB. My goal is to justify the proposition that the object-oriented approach is feasible for the EWIRDB by providing a conceptual design of a representative portion of the database - a portion that adequately reflects the nature of electronic warfare data. Diagrams are used at every stage to codify the conceptual design. A description of the conceptual design symbology must first be addressed.

The absence of a standardized OODM introduces some variation in its diagrammatic representation. However, most of the symbology adopted in this thesis for the conceptual design of the EWIRDB is commonly used. Possible exceptions are those notations corresponding to abstractions such as covering and aggregation. Variations aside, the consistent use of an adequately-expressive symbology is all that matters.

The symbology used in the conceptual design of the EWIRDB are shown in Figure 7. The inheritance abstraction as it appears in Figure 7 includes some detail not previously addressed. The concept of *overlapping inheritance* stipulates that an object of the superclass (generalization class) may be a member of more than one subclass of the specialization. *Disjoint inheritance* states that an object of the superclass may be a member of at most one subclass of the specialization. Regardless of the type, however, each inheritance hierarchy in the conceptual design of the EWIRDB is a *total specialization*. This idea states that every object of a superclass must be a member of some subclass in the given inheritance hierarchy[2].

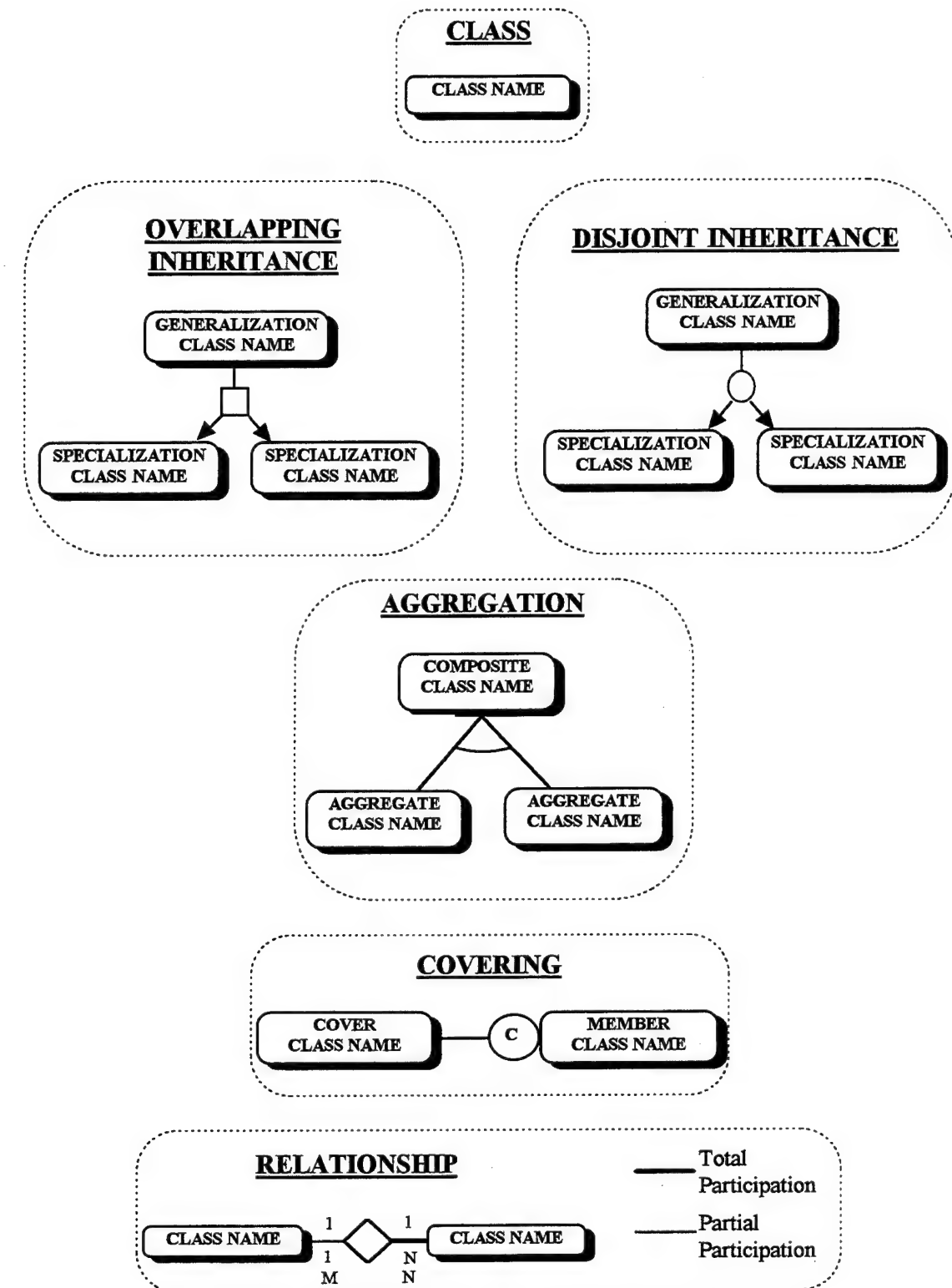


Figure 7. Conceptual Object-Oriented Design Symbology

The representation of relationships requires amplification as well. Figure 7 includes a description of the *participation constraints* in relationships between the objects of participating classes. A *total participation constraint* indicates that for the class of objects whose participation in a given relationship is total, the very existence of that class of objects depends on its participation in the relationship. For example, in a relationship between common entities such as transportation vehicles and license plates, the participation of license plates would be total; license plates are unnecessary if there are no vehicles to license. Ergo, the existence of license plates depends on the relationship between cars and license plates. A *partial participation constraint*, in contrast, states that all objects of a particular class need not participate in a given relationship. In a relationship between married couples and children, for instance, not all married couples have children. The participation of married couples in the relationship is therefore partial. Participation constraints are an important aspect of conceptual modeling. They further characterize the nature of data relationships.

A. A GLOBAL SCHEMA

The EWIRDB was described earlier (Figure 1) as the administrative merging of data from three contributory sources. Now, a global and object-oriented view of the merged structure of the EWIRDB is provided in Figure 8 with the use of aggregation semantics. The "big picture" object-oriented view of the EWIRDB in Figure 8 is largely administrative. It may at first seem strange to proceed in this manner, to initially approach the modeling task from an administrative rather than technical point of view, especially in light of the technical nature of emitter data. But this approach is valid. As explained in Chapter 1, the data items that describe an emitter retain the formatting particular to the database from which they were contributed. In a global view, source-specific groupings of data items are assigned group-specific administrative labels. A design that proceeds within an administrative context preserves these important associations.

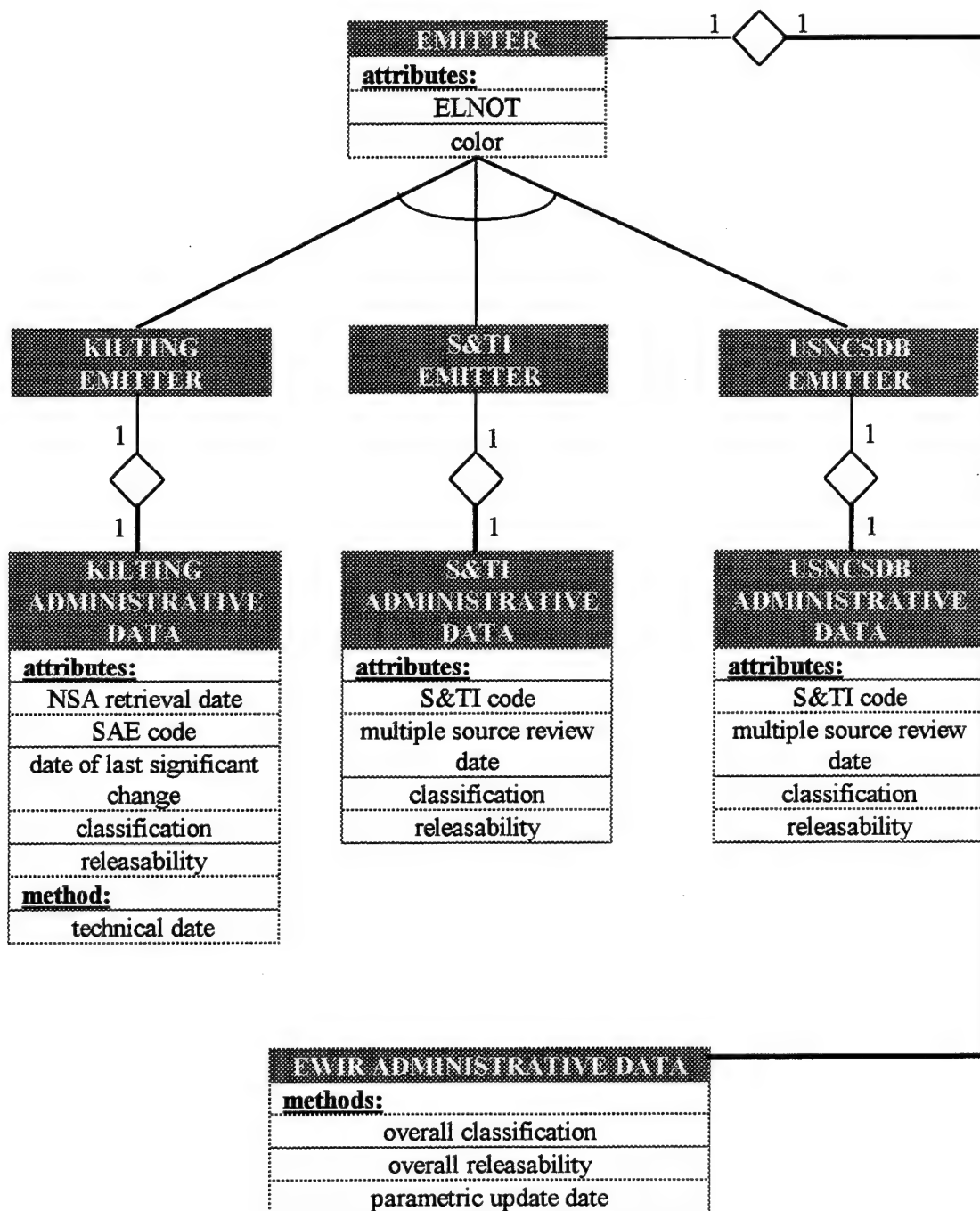


Figure 8. The Conceptual Schema: A Global View

As a result of the merging, the data items contributed by each source to describe a particular emitter may overlap. Moreover, each source may contribute multiple value entries for a given data item. But the identity of each data entry remains intact. Multiple entries for a given data item are not "fused" together to form a single EWIRDB entry. Each data item remains separate and distinct, in a form that is suggestive of its source. Approaching the conceptual design from an administrative bias thus ensures that the overall structure of the database as a collection of emitter data from multiple sources will be accurately reflected in the object-oriented schema.

In Figure 8, the aggregates **KILTING EMITTER**, **S&TI EMITTER**, and **USNCSDB EMITTER** combine to form the composite **EMITTER** class of objects. This aggregation precisely models the multi-source structure of the database. As the composite, an **EMITTER** object represents the merging of all data for a given emitter. Each aggregate, on the other hand, represents a source-specific portion of the data in the composite. The aggregate **KILTING EMITTER** encapsulates Kilting technical data contributed to the EWIRDB for a given emitter. The **S&TI EMITTER** aggregate encapsulates the technical data contributed from S&TI centers, and the **USNCSDB EMITTER** aggregate encapsulates USNCSDB data for a given emitter.

With aggregation semantics, emitter parametric data may be reasoned about on two levels of abstraction: in the composite, dealing with all available data, or on the aggregate (component) level, where the data from a particular contributory source is considered singularly. This adds a degree of flexibility in the manipulation of data that may not be achievable in more conventional models. Further, categorizing emitter data by source is appropriate because it allows the drawing of relationships between source-specific administrative data and the aggregates themselves. In Figure 8, each aggregate participates in a 1:1 relationship with an administrative-data class of objects; **KILTING EMITTER** with **KILTING ADMINISTRATIVE DATA**; **S&TI EMITTER** with **S&TI ADMINISTRATIVE DATA**; and **USNCSDB EMITTER** with **USNCSDB ADMINISTRATIVE DATA**. The participation of each administrative data class in its

respective 1:1 relationship is total because the existence of an administrative data class depends solely on the viability of the relationship. If, for instance, the Kilting database contributed no data to the EWIRDB for a given emitter, then for that given emitter, the **KILTING EMITTER** class of objects would be undefined. In effect, **KILTING EMITTER** would be non-existent, as would any relationship in which it participated. In this example, the existence of the **KILTING ADMINISTRATIVE DATA** class of objects would be meaningless as well.

As mentioned in Chapter I, the formatting for S&TI and USNCSDb data are the same. The attributes in the related administrative data classes are also the same. The attribute values, however, are likely different between the two classes. This does not rule out the possibility that some or all of the values may be identical. But the possibility, likely or not, that the attribute values may differ depending on the source necessitates the appearance of the same attributes in both classes. For the same reason, the attributes **classification** and **releasability** are duplicated in all three administrative data classes. This seems to contradict object-orientation, wherein commonality is factored out among similar classes to form a superclass. However, because the attribute values may possibly be different from class to class, the common attributes, by virtue of their values, still function to differentiate the classes. In these particular situations, the semantics of generalization simply do not apply and the same attribute values may appear in each class.

The **EWIRDB ADMINISTRATIVE DATA** class contains methods to extract information from the source-specific classes in the schema. These methods retrieve administrative data for a given emitter that in turn define the administrative state of all merged emitter data. An object of the class **EWIRDB ADMINISTRATIVE DATA** may reflect data retrieved from more than one class of the schema. The method **overall classification** returns the highest classification from among the source-specific administrative data classes; it defines the classification or the composite classification for a given emitter. Although not at present an attribute explicitly accounted for in the EWIRDB, the method **overall releasability** was included to satisfy the requirement that

“...The releasability and handling caveats reflect a merger of the three sources...”[1] This method, like the first, returns the most stringent of the releasability instructions and thus defines the releasability for the data of a given emitter when the data are considered in the composite. The method **parametric update** searches through all the class attributes in the database for a given emitter and returns the latest data update date. The effect is an **EWIRDB ADMINISTRATIVE DATA** class that describes the composite **EMITTER** class of objects. **EWIRDB ADMINISTRATIVE DATA** therefore participates in a 1:1 relationship with **EWIR EMITTER**. And like the source-specific administrative data classes, its participation in the relationship is total.

In Figure 8, the attribute **ELNOT** in the **EMITTER** class is a kind of social security number for emitters. It uniquely identifies an emitter, or more precisely, the signal that is characteristic of an emitter. **ELNOT** is an important attribute because it is the primary means of emitter identification, and may often be the launch point for EWIRDB queries. The attribute **color** is an appropriate addition to **EMITTER** because it describes, in general, an emitter's role in terms of friendly or hostile use. The choice of attribute values are “blue” for those emitters aboard US military platforms, “blue/gray” for those originally in US production that were legitimately transferred to Rest of World (ROW) countries (non-US, non-Communist), “gray” for emitters aboard non-Communist country platforms, and “red” for emitters produced by Communist countries [guide]. The attribute **color** thus provides a big picture look at an emitter. Because it is not a source-specific characteristic, it is best placed in the composite class.

The global, object-oriented view of the EWIRBD presented in Figure 8 incorporates all the data elements contained in the S00 and S01 records in the TERF output. The *S&TI Code* found in S01 records (Figure 5) is included in both the **S&TI ADMINISTRATIVE DATA** and **USNCSDB ADMINISTRATIVE DATA** classes. It therefore applies to all assessed data encapsulated within an instantiation of **S&TI EMITTER** or **USNCSDB EMITTER**. The duplicate *S&TI Code* entry found in S03 records (Figure 5) is removed from any further consideration.

Object-orientation eliminates the need for S02 branch information. The S02 data element *Technical Date* (Figure 5), however, specific to Kilting emitter data, is included as a method in the **KILTING ADMINISTRATIVE DATA** class. Similar to the method **parametric update**, this method returns a date that indicates the latest update to emitter data, but applies to smaller, more specific groups of data. These groups are collections of generally related data elements, referred to in this thesis as logical groupings. Logical groupings are introduced in section B and elaborated in section C.

The benefit of object-orientation is a more coherent and intuitive design. Now, for a given emitter, administrative and technical emitter data are encapsulated within the **EMITTER** class via aggregation, relationships, and inheritance. To this point in the conceptual design, particularly from the administrative point of view, the presentation of data is clearer than that found in the parametric tree-TERF model.

B. THE EMITTER SCHEMAS

The next step in the development of the conceptual design focuses on the technical aspect of emitter data and addresses the data encapsulated within the classes **KILTING EMITTER**, **S&TI EMITTER**, and **USNCSDB EMITTER**.

To reiterate, the conceptual designs presented in this section are based on portions of the EWIRDB. These portions are sufficiently representative of the entire database and accurately reflect the nature of EW data. Because the focus of this section is the overall organization of emitter parametric data, the detail of object structure and behavior is omitted. (Specific class attributes are provided in Chapter V as part of the logical design.) This does not, however, take away from the intended semantics, and the schematics reveal the utility of the object-oriented approach in providing a unified and intuitive picture of emitter parametric data.

1. The Kilting Emitter Class

The overall configuration of the data encapsulated within the aggregate **KILTING EMITTER** class is depicted in Figure 9. An emitter object is not described as the composite of its component parts, i.e., an aggregation. Modeled as an aggregation, the analysis of complex EW emitters could then be one of a hardware-oriented divide-and-conquer. An overall performance assessment could be made based on the intermediate results obtained in the evaluation of the hardware components. But the hardware components themselves are not central to the discussion of EW. For the purposes of the EWIRDB, hardware components are only important in that they have some effect on, or participate in the generation of, a given signal. The signal itself is pivotal in the analysis - not the hardware. This is reflected in the design shown in Figure 9. Rather than being exposed hardware component by hardware component, the **KILTING EMITTER** class of objects is instead related to several *logical groupings* of data, all of which are signal-based in their description of emitter performance.

KILTING EMITTER participates in a one to many relationship with **ANTENNA**, a class that encapsulates a logical grouping of antenna-signal data. A single emitter may contain one or more antennas, each of which may have a different function or produce a different effect on a signal. However, antenna hardware is not explicitly addressed within the antenna-data grouping. Modeling the relationship between **KILTING EMITTER** and **ANTENNA** as one-to-many is not intended to treat this portion of EWIRDB data as hardware oriented, although this may be a collateral effect. More important is the effect of any given antenna on an emitter's signal. The one-to-many relationship reflects the fact that there may be multiple antennas, or multiple versions of antenna data for a particular emitter, depending on the number of antennas and the availability of information on each. The antenna data grouping is given more detailed treatment in section C.1.

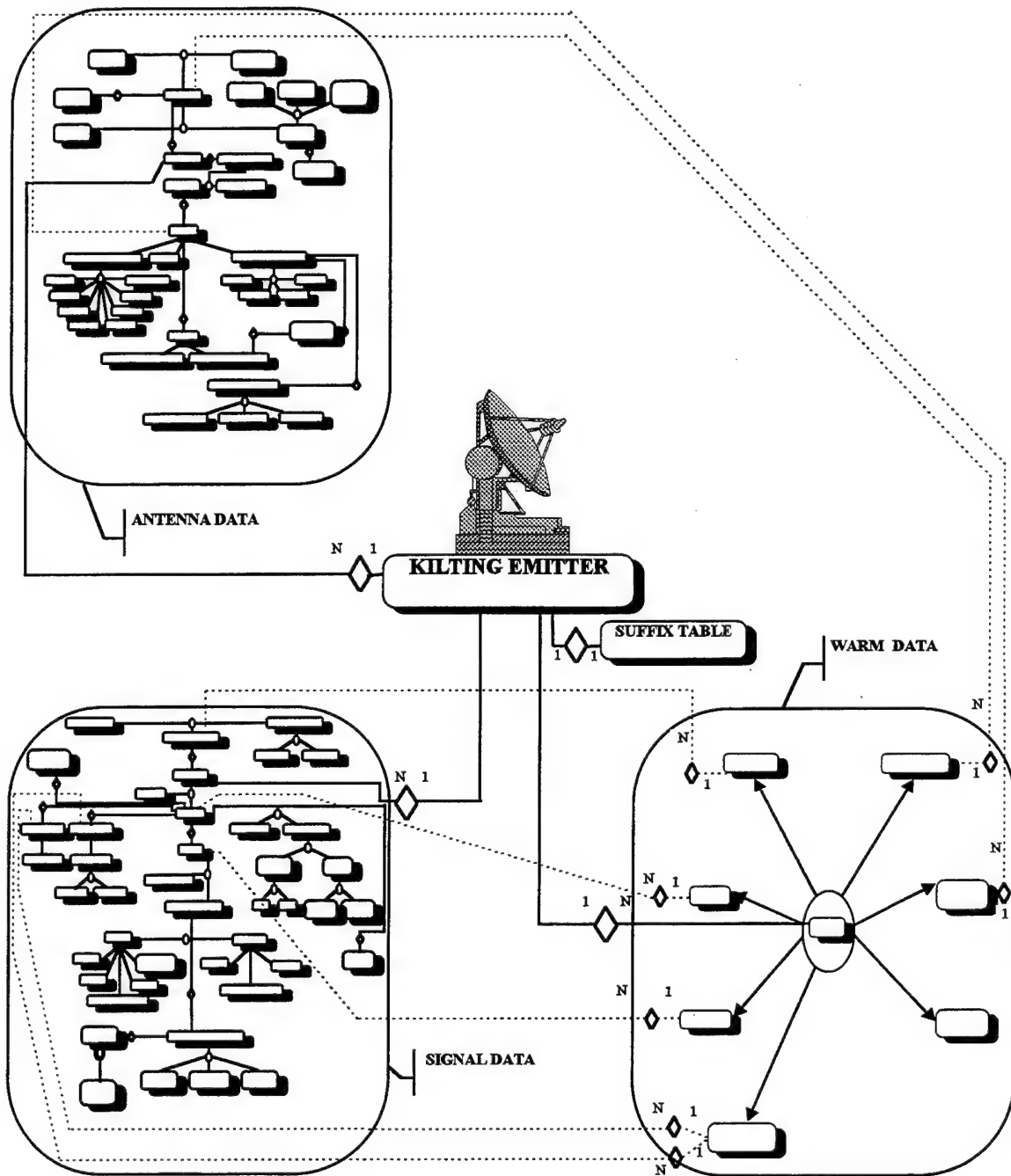


Figure 9. The Conceptual Schema: Kilting Emitter Data

KILTING EMITTER participates in a one-to-many relationship with the class **SIGNAL**, perhaps the most important grouping of data in identifying an emitter and its signal signature. The one-to-many relationship indicates that an emitter's identifying signal is subject to variation. A change in the configuration of the emitter's controls, for example, causes a variation in the signal. Therefore, an emitter's signal may behave differently, with respect to fundamental signal characteristics, depending on the employment of the emitter. Signal characteristics are described in section C.2.

KILTING EMITTER also participates in a one-to-many relationship with the **WARM (Wartime Reserve Mode)** class, which encapsulates those signal characteristics likely to be encountered only when an emitter is in a wartime reserve mode. A single emitter may have from zero to many such special modes. Wartime reserve modes are those emitter capabilities, deliberately held in reserve, that differ from or exceed normal-use capabilities. **WARM's** are used exclusively in emergency or wartime scenarios to counter attempts to exploit the perceived weaknesses in an emitter's performance. A sound assessment or a foreknowledge of the **WARM's** employed by an enemy can be a huge advantage in the prosecution of EW. **WARM** data is therefore an important aspect of the **EWIRDB**.

To provide for a simplified diagrammatic layout, the **WARM** class is surrounded by a circle to represent the existence of a disjoint inheritance hierarchy. **WARM** data is examined more closely in section C.4.

Finally, **KILTING EMITTER** objects have a one-to-one relationship with the **SUFFIX TABLE** class of objects. The suffix table as it currently exists in the **EWIRDB** describes complex emitter mode combinations in concise fashion. Knowledge of these combinations allow **EWIRDB** analysts to establish emitter performance patterns and mode usage tendencies. The suffix table is thus an important tool that helps the analyst to discriminate between signals and ultimately associate a signal to an emitter. It is examined more closely in section C.5.

2. The Association of an Emitter to its Signal

Associating a unique signal to its emitter is a difficult modeling problem, object-oriented or otherwise. The association is characterized by an ELNOT that uniquely identifies an emitter that is uniquely identified by its signal signature. (ELNOT is an attribute encapsulated within the **EMITTER** class shown in Figure 8.) More precisely, the ELNOT is "assigned to each noncommunications emission for collection guidance and reporting purposes." [1] Thus, the uniqueness of the ELNOT, assigned to noncommunications emissions, implies a one-to-one relationship from signal to emitter, or equivalently from emitter to signal. This modeling is easy to reason about in theory, but in application, hard to achieve. In Figure 9, the general organization of the parametric data describes an emitter by its signal attributes within the context of antenna-induced effects (**ANTENNA DATA**), signal characteristics in general (**SIGNAL**), reserve modes (**WARM**), and combinations of modes (**SUFFIX TABLE**). While this design provides a comprehensive view of signal-based parametric data, the one-to-one nature of the relationship between emitter and signal becomes obscured. Although the data are more semantically meaningful when described within the logical groupings, the effect is a partitioning of the data. Consequently, a relationship must be developed between the emitter (**KILTING EMITTER**) and each partition. Several relationships then exist to describe the relationship between emitters and signals. Not all, however, are one-to-one; **KILTING EMITTER** participates in a one-to-many relationship with **ANTENNA DATA**, **SIGNAL DATA**, and **WARM DATA**. The end result is an association between an emitter and its signal, but the uniqueness of the relationship is directly modeled only through the use of the ELNOT. The ability of an emitter to vary its signal characteristics - and effectively produce more than one signal - makes it more difficult to visualize the one-to-one nature of the relationship between emitter and signal, and therefore between ELNOT and emitter. The one-to-one relationship remains intact, but is perhaps more identifiable because of the existence of the ELNOT.

3. The S&TI and USNCSDb Emitter Classes

As discussed in Chapter I, the S&TI community produces performance assessments based on an exhaustive search of all available information. These assessments are particularly useful in developing an understanding of an emitter's receiver capabilities. USNCSDb data, derived from equipment specifications, also includes receiver performance data. Similar in all other design aspects, the USNCSDb and S&TI conceptual designs shown in Figures 10 and 11 are the same.

In contrast, the **KILTING EMITTER** schema in Figure 9 does not contain receiver data because Kilting data reveals nothing about receiver performance. Kilting data are obtained from the direct analysis and measurement of emitter signals following signal intercept. An emitter's receiver, however, produces no obvious observable effect. The class **RECEIVER**, which encapsulates the logical grouping of receiver data in both Figures 10 and 11, is similar to the **ANTENNA** class in that the information it presents is important in describing the effect of hardware on any given signal. **RECEIVER**, however, encapsulates data that tends to be more hardware-oriented. The receiver's function is to accept a signal, process it, and then relay signal information to a display. A receiver's manipulation of a signal is strictly internal and does not directly produce an effect that is visible in the atmosphere. The internal function of receivers in processing signal data is therefore best described within the context of hardware components.

The one-to-many relationship between the emitter class and **RECEIVER** in Figures 10 and 11 convey the idea that an emitter may consist of one or more receivers. The receiver data grouping is given more detailed treatment in section C3.

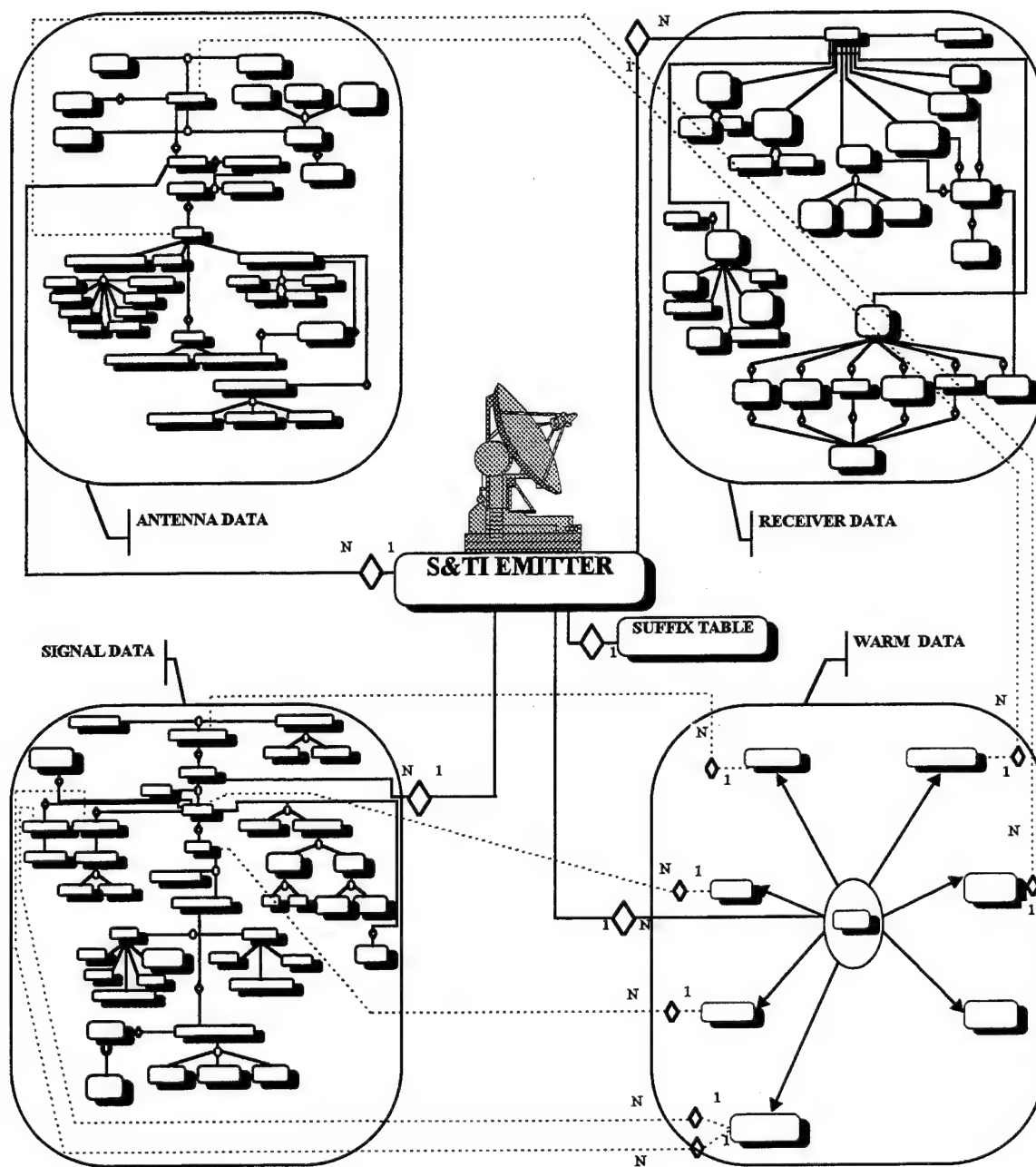


Figure 10. The Conceptual Schema: S&TI Emitter Data

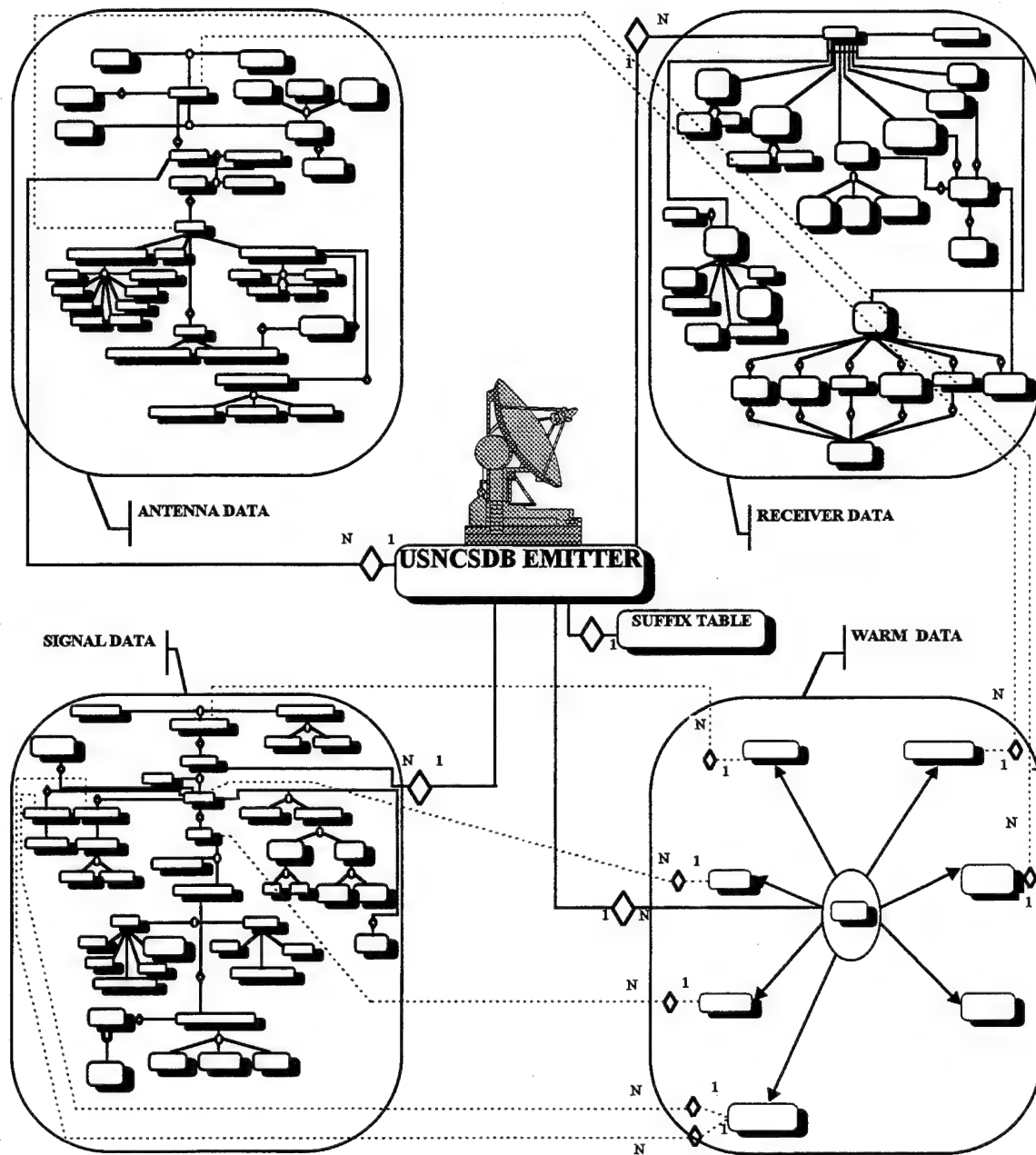


Figure 11. The Conceptual Schema: USNCSDB Emitter Data

C. THE SCHEMAS WITHIN LOGICAL GROUPS

In this section I present a more detailed view of the data contained within the logical data groupings described in the previous section. Logical groupings, like the subfiles that currently exist in the EWIRD, encompass logically-related data elements. But the schemas depicted in this section reinforce the notion that the OODM provides for data semantics previously unachievable in the EWIRDB. Emphasis is placed on the schema design; complete technical descriptions of each data class are provided in [8] and [9]. Supplemental information is provided in [10] and [11].

1. Antenna Data

Figure 12 is an enlargement of antenna-related signal data. It represents a substantial improvement over the semantically limited hierarchical tree representation of antenna data discussed in section I.B.1.b.

Specifically, an antenna may exhibit a polarization and a particular radiation pattern, as described by the one-to-one relationship between **ANTENNA** and **POLARIZATION**, and by the one-to-one relationship between **ANTENNA** and **RADIATION PATTERN**. Two disjoint hierarchies branch out from the **POLARIZATION** class. One addresses the orientation of the electromagnetic wave, specializing the polarization as either linear or circular/elliptical. The other describes the variation of the polarization as either fixed or variable. Thus, using the tools of the OODM, the four possible polarization combinations - fixed-linear, fixed-circular/elliptical, variable-linear, variable-circular/elliptical - are captured intuitively in the schema. An antenna's cross polarization characteristics are now correctly modeled in the one-to-one relationship between **POLARIZATION** and **CROSS POLARIZATION**. No longer are cross polarization characteristics confused with those that determine an antenna's design wave orientation or its polarization variation properties. Moreover, access to data

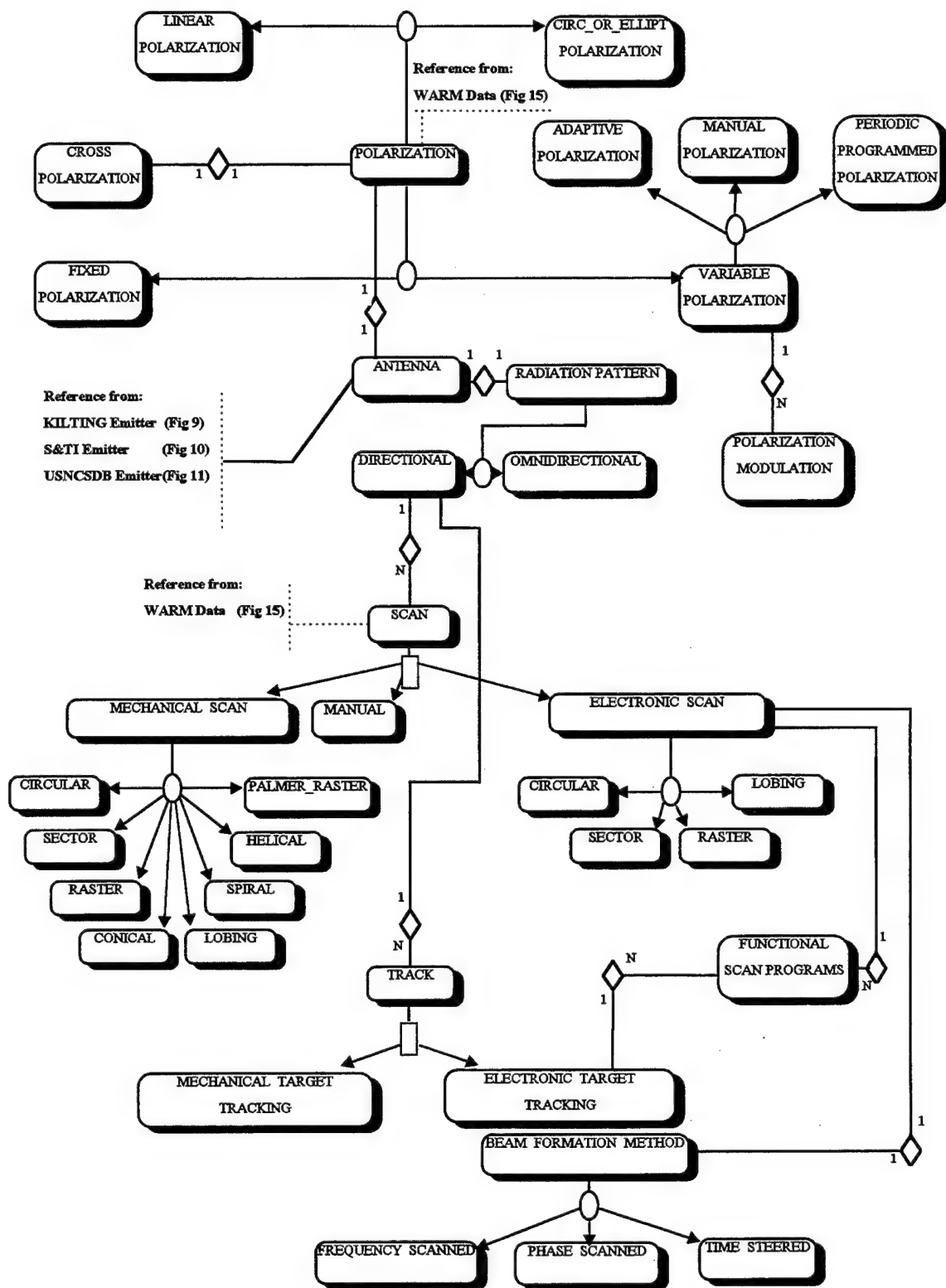


Figure 12. The Conceptual Schema: Antenna Data Enlargement

concerning an antenna's polarization also guarantees access to data concerning the antenna's cross polarization, via the relationship.

The same is now true for antenna data connected with the **VARIABLE POLARIZATION** class. The classes **ADAPTIVE POLARIZATION**, **MANUAL POLARIZATION**, and **PERIODIC PROGRAMMED POLARIZATION** in the hierarchies are clearly specializations, or types of variable polarization. The class **POLARIZATION MODULATION**, possibly mistaken for a type of variable polarization in the parametric tree model, is instead related to, and therefore descriptive of, **VARIABLE POLARIZATION** via the one-to-many relationship.

The remainder of Figure 12 provides a straightforward representation of other aspects of an antenna's functionality. An antenna may radiate directionally or omnidirectionally. If the antenna is directional, then it is associated with one or more scanning techniques. The antenna scan data is further refined within the specialization's subordinate to the mechanical and electronic scan classes. In addition, an electronically scanning antenna may be controlled by one or more scan programs, and employs a beam formation method to effect its scanning movement. A directional antenna also performs a tracking function, which may include simultaneous mechanical and electronic tracking. Finally, the features of electronic scanning are largely determined by one or more functional scan programs.

Figure 12 represents some of the salient aspects of antenna data in a way that is understandable to both expert and novice. This depiction of antenna data, in the form of the OODM, is clearly superior to that of the arbitrary tree model.

2. Signal Data

Figure 13 depicts an enlargement of the logical grouping of signal data and addresses the complexities of signal transmission in a natural and intuitive way. The information contained in this grouping details fundamental signal characteristics.

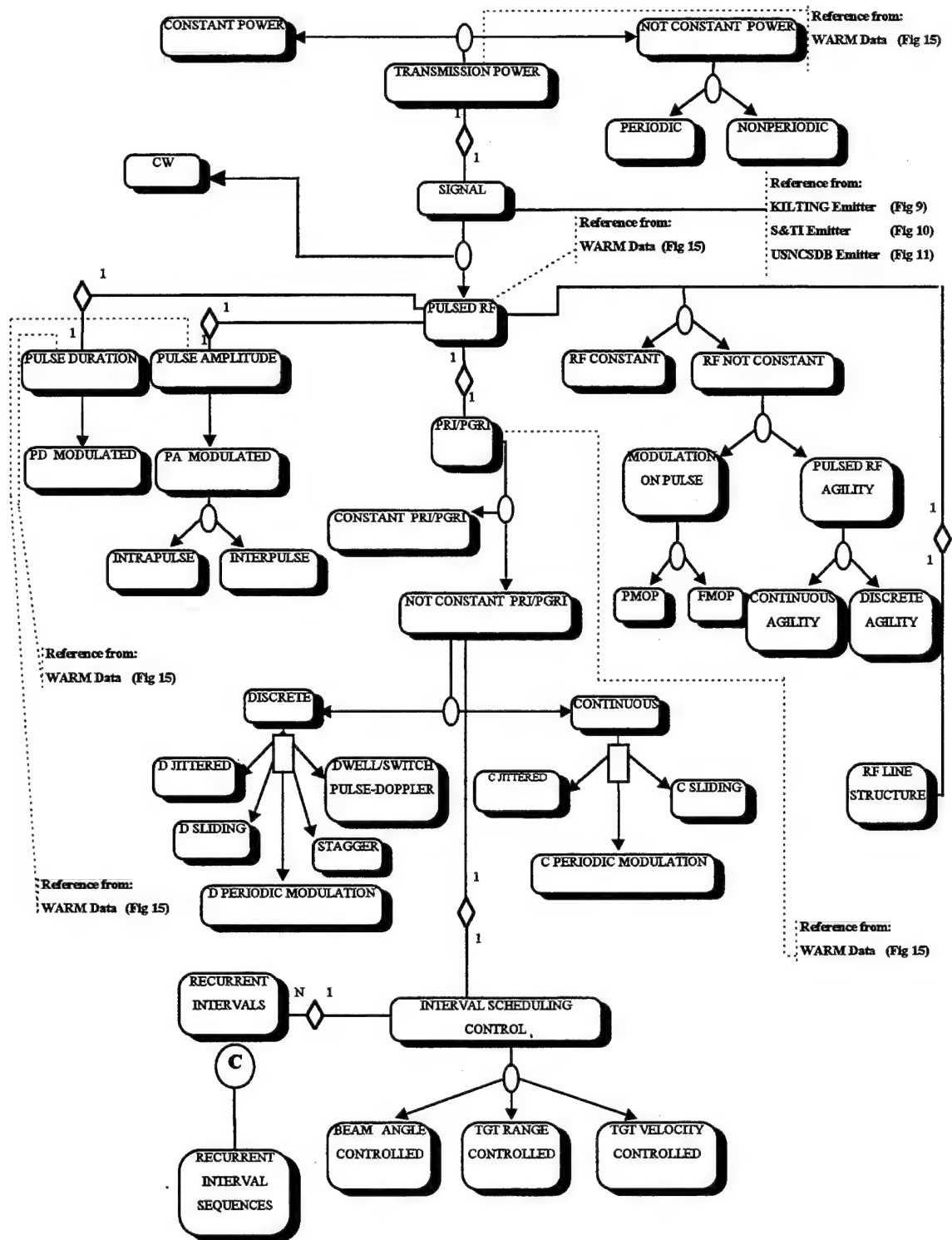


Figure 13. The Conceptual Schema: Signal Data Enlargement

Any given signal is generated with a certain power that is either constant or variable in nature. The object-oriented representation exactly matches these semantics. **SIGNAL** participates in a one-to-one relationship with **TRANSMISSION POWER**, the generalization of the specialization classes, **CONSTANT POWER** and **NOT CONSTANT POWER**. The **SIGNAL** class is the root of the inheritance hierarchy that spawns the **PULSED RF** (Pulsed Radio Frequency) and **CW** (Continuous Wave) specialization classes. **PULSED RF** is the basis of the conceptual schema in Figure 13; it is the starting point in the modeling of basic signal characteristics such as pulse duration, pulse amplitude, pulse repetition interval (PRI) and pulse group repetition interval (PGRI), and frequency (RF). (CW signal characteristics are represented within the class **CW** but are not examined any further.)

For a given occurrence of **PULSED RF** there exists a one-to-one relationship with the classes **PULSE DURATION**, **PULSE AMPLITUDE**, **PRI/PGRI**, and **RF LINE STRUCTURE**. These classes characterize in detail the nature of a given signal pulse. **PULSED RF** is a generalization class in the inheritance hierarchy that refines the description of a pulsed signal's carrier frequency. The basis of specialization is the constancy of the carrier frequency. A given pulsed signal therefore belongs to either the **RF CONSTANT** class or **RF NOT CONSTANT** class. A subordinate hierarchy rooted at **RF NOT CONSTANT** further describes the nature of the variation in the carrier frequency.

Objects in the classes **PULSE DURATION** and **PULSE AMPLITUDE** may be static or variable, as indicated by the specialization classes **PD MODULATED** and **PA MODULATED**, respectively. Both are single specializations. If, for instance, an object of the class **PULSE DURATION** is not modulated, then there is no information outside of the class **PULSE DURATION** that is required to describe that object. If the intent is to describe a modulated pulse duration, then additional or specialized data is required, and an object of class **PD MODULATED** would be instantiated.

Objects of the class **PRI/PGRI** belong to either the **CONSTANT PRI/PGRI** specialization class or the **NOT CONSTANT PRI/PGRI**, depending on the pulse-to-pulse changes in pulse interval. A member of the **NOT CONSTANT PRI/PGRI** reflects interval changes that are either discrete or continuous. The classes **DISCRETE** and **CONTINUOUS** are themselves generalizations in overlapping inheritance hierarchies. Additionally, a pulsed signal whose pulse repetition interval is not constant exhibits the characteristics of some type of interval scheduling control. A one-to-one relationship therefore exists between **NOT CONSTANT PRI/PGRI** and the class **INTERVAL SCHEDULING**. An interval scheduling control induces one or more recurrent pulse repetition intervals. The schema therefore includes a one-to-many relationship between **INTERVAL SCHEDULING CONTROL** and the class **RECURRENT INTERVALS**. The **RECURRENT INTERVALS** class is important in its description of recurrent interval sequences; it may be thought of as a higher level abstraction for an arrangement of interval sequences. Moreover, it becomes meaningless as an abstraction without the existence of interval sequences. Viewed in this way, a mapping may be developed between recurrent interval and recurrent interval sequences. In Figure 13 this mapping is represented as a covering; cover class **RECURRENT INTERVALS** covers the member class **RECURRENT INTERVAL SEQUENCES**.

3. Receiver Data

Aggregation semantics model the hardware-orientation of the receiver data. In Figure 14, the class **RECEIVER** is the "outermost" composite in a nested aggregation wherein some of a receiver's aggregates are themselves composites that are composed of aggregates. Objects that belong to the classes **IF PREAMPLIFIER** and **IF AMPLIFIER**, for example, are aggregates of objects of the class **INTERMEDIATE FREQUENCY SECTION**. Objects of the class **INTERMEDIATE FREQUENCY SECTION** are, in turn, aggregates of objects of the class **RECEIVER**.

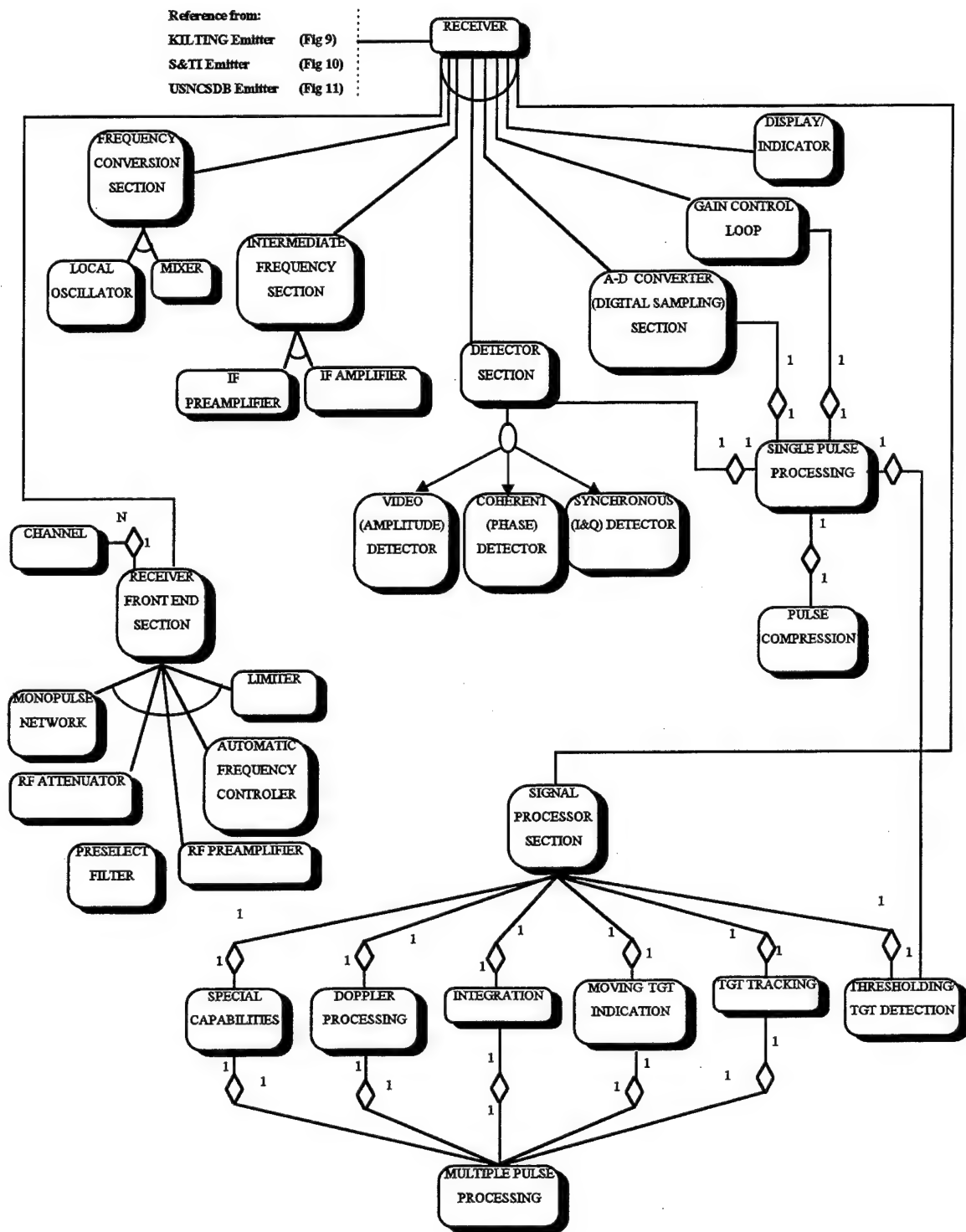


Figure 14. The Conceptual Schema: Receiver Data Enlargement

Thus, **RECEIVER** represents the sum total of all components that function together to perform the receiver task. More precisely, receiver functionality and component functionality, not hardware, is the basis of the aggregation. The specifics of the hardware is important only in drawing boundaries between functional sections of components that are common to all receivers. And despite hardware differences, all receivers may be modeled in this way because of a similar functionality. This is a logical and natural representation of the data. The receiver portion of an emitter may now be reasoned about in general terms as a singular unit and, or exposed in more detail as the aggregation, or nested aggregation, of distinct functional sections.

Many of the actions performed by a receiver are described as either single pulse processing or multiple pulse processing. These labels can be assigned to receiver processes, within the setting of aggregation semantics, as shown in Figure 13. In the schema, applicable object classes participate in one-to-one relationships with **SINGLE PULSE PROCESSING** objects or **MULTIPLE PULSE PROCESSING** objects. However, both the **SINGLE PULSE PROCESSING** and **MULTIPLE PULSE PROCESSING** classes exist solely to provide the capability to access receiver-signal information from a single pulse/multiple pulse processing bias. Their primary purpose is navigational. These two classes are descriptive of receiver data in name alone.

Multiple one-to-one relationships originate from the **SIGNAL PROCESSOR SECTION** class. The other participating classes - **SPECIAL CAPABILITIES**, **DOPPLER PROCESSING**, **INTEGRATION**, **MOVING TGT INDICATION**, **TGT TRACKING**, and **THRESHOLDING/TGT DETECTION** - encapsulate data that describe the functionality of a receiver's signal processor section. The choice to use reference relationships instead of aggregation relationships is based on the composition of the EWIRDB data. In general, as signal processing is addressed with an increasing level of detail with respect to functionality, hardware differences among receivers tends to become more pronounced. In other words, as the granularity of data increases, receivers may still be described in terms of common functionality, but tend to be less alike in

hardware. Functionality is therefore less prone to be cast in the light of hardware as the data become more detailed. The other classes are not so much parts of **SIGNAL PROCESSOR SECTION** as they are descriptors of the types of actions performed in that section. Aggregation semantics become less applicable; reference relationships better model the nature of this interaction.

4. WARM Data

The design of Figure 15 echoes previously introduced elements of the conceptual schema. For example, the class **POWER ECCM** participates in a one-to-many relationship with **TRANSMISSION POWER** from Figure 12. As will be found in the logical design, this relationship indicates that a WARM mode affecting signal power, or an object of the class **POWER ECCM**, is essentially a new instantiation of the class **TRANSMISSION POWER**. WARM modes that are not variations of existing data classes fall within the class **OPERATIONAL ECCM**.

The inheritance hierarchy in Figure 15 is disjoint; any given object of the **WARM** class is a member of only one specialization class. However, an emitter may exhibit multiple WARM modes, as modeled in the one-to-many relationships between the classes **KILTING EMITTER** and **WARM**, **S&TI EMITTER** and **WARM**, and **USNCSDB EMITTER** and **WARM**.

This approach to the modeling of WARM data does away with the need to account for the *Reserve Mode* entry found in S03 records (Figure 5).

D. THE PARAMETRIC DATA CLASS

As discussed in section III.A.1.b, complex attributes support objects as attribute values. Therefore, in the case of complex attributes, the "type" of a given attribute is

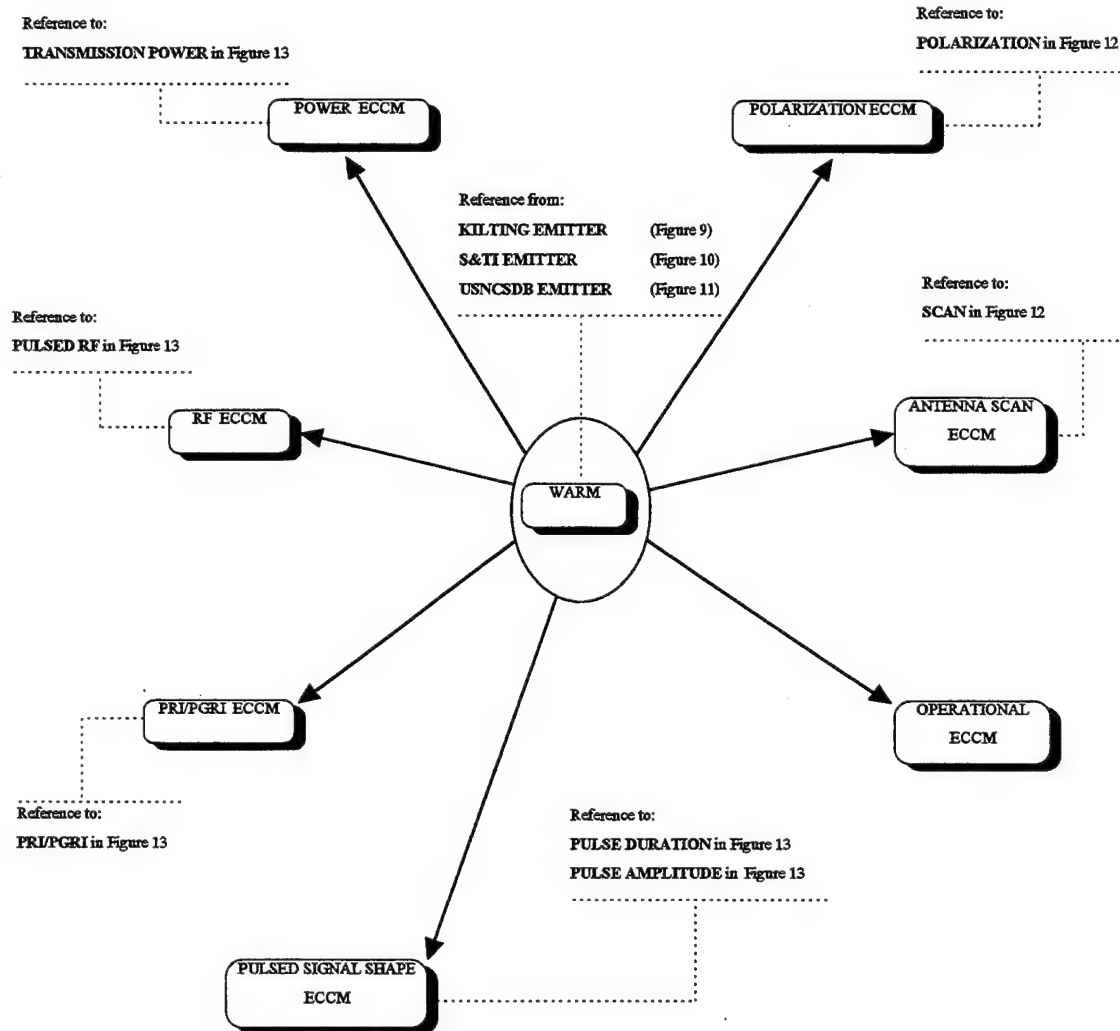


Figure 15. The Conceptual Schema: WARM Data Enlargement

equivalent to the particular object class from which that object value was instantiated. Further, complex attributes may reflect an arbitrarily deep or recursive nesting of objects.

All complex attributes in the object-oriented design of EWIRDB, regardless of the depth of nesting, ultimately lead to objects of the class **PARAMETRIC DATA**, the focus of this section. The **PARAMETRIC DATA** class itself exhibits a nesting of objects that incorporates the semantically-useful data elements of the S03, S04, and S05 records.

The **PARAMETRIC DATA** class and the data encapsulated therein is depicted in Figure 16. **TEXTUAL DATA** and **NUMERIC DATA** are specializations of **PARAMETRIC DATA**, and intuitively indicate whether the parametric data entry for a given attribute is text-based or numerical. Numerical parametric data are either single-valued or range-valued as expressed in the specialization classes **SPECIFIC VALUE** and **VALUE RANGE**.

In the EWIRDB, comments are used to further characterize parametric data values. **PARAMETRIC DATA** thus participates in one-to-one relationship with the class **DATA COMMENT**. The participation of **DATA COMMENT** in the relationship is total; a parametric data entry must first exist before a corresponding comment can be made, but not all parametric data entries must be commented. If a parameter is assessed, then a related comment must also include the comment classification. This is depicted in the specialization class **ASSESSED DATA COMMENT**. Comment data and the inheritance hierarchy directly subordinate to the **PARAMETRIC DATA** class are enclosed within the dotted line in Figure 16. Together they constitute the core of EWIRDB parametric data.

On the global scale, each emitter is linked to emitter-specific administrative data; on a smaller scale, each class attribute is associated with the attribute-specific administrative data associated with the S03, S04, and S05 records. The attribute-specific administrative data identifies data references and highlights important descriptive information. As indicated in Figure 15, the format of this data is source-dependent. **ORIGINAL SOURCE DOCUMENTATION** includes those attributes common to both

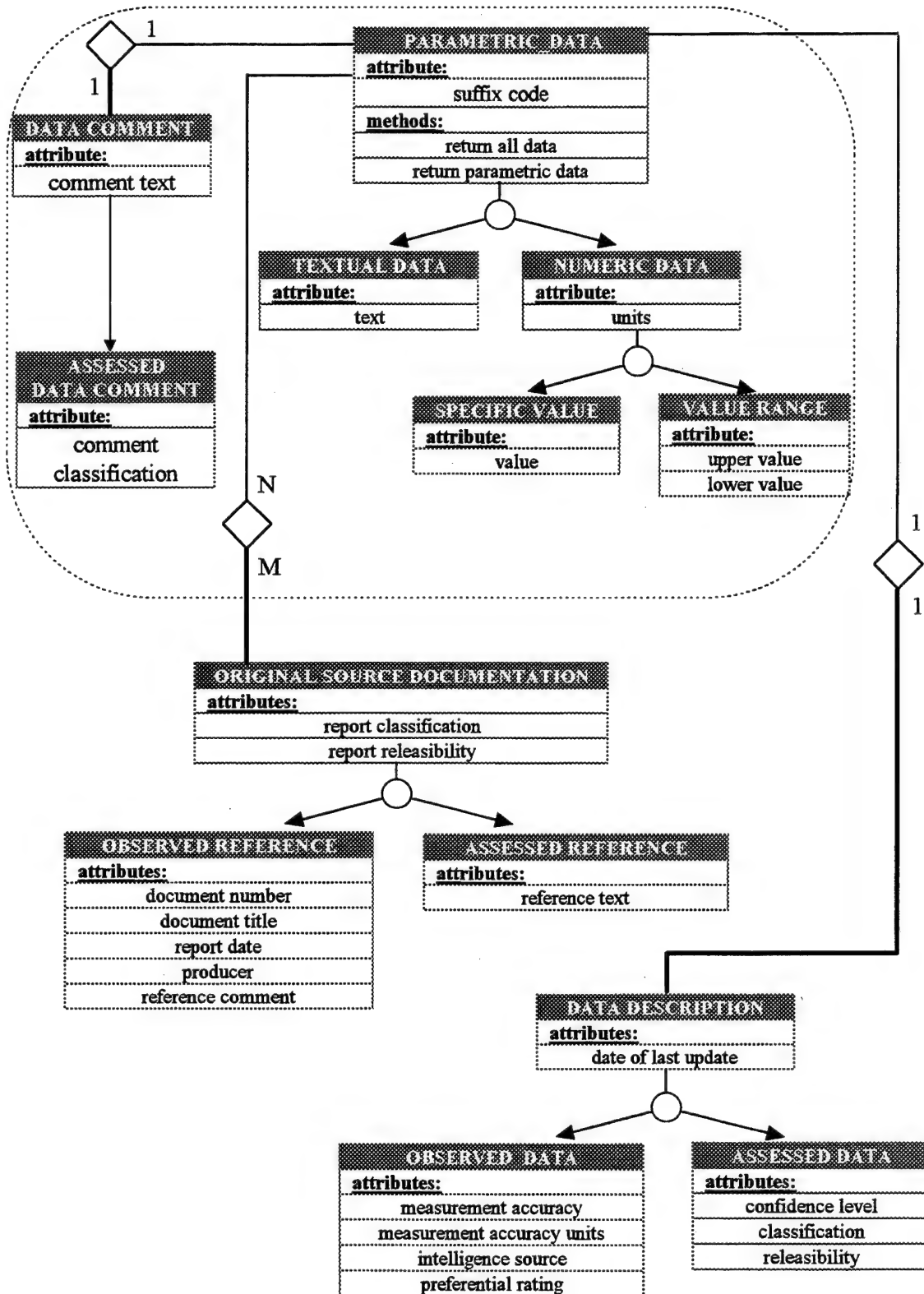


Figure 16. The Conceptual Schema: Parametric Data

formats, report classification and report releasability. Formatting distinctions are made within the specialization classes **ASSESSED REFERENCE** and **OBSERVED REFERENCE**. Attribute values are further characterized in the **DATA DESCRIPTION** class by **date of last update**. (The method **parametric update date** in the class **EWIRDB ADMINISTRATIVE DATA** (Figure 9) accesses **date of last update** information throughout the database and returns the most recent value for a given emitter.) Source-dependent characteristics that generally describe the soundness and accuracy of a given attribute value are addressed in the specialization classes **ASSESSED DATA** and **OBSERVED DATA**.

Two methods are specified in the **PARAMETRIC DATA** class: **return all data** and **return parametric data**. For a given attribute, **return all data** will reply with all available data - the actual parametric data as well as the associated administrative data. **return parametric data** will yield only the data enclosed within the dotted line in Figure 15 for a given attribute. One attribute is specified as well, **suffix code**, a label for the given attribute as it appears in the suffix table.

Thus, all useful data items from the S03, S04, and S05 records, with the exception of suffix table information, are nested within the **PARAMETRIC DATA** class of Figure 15. Object-orientation eliminates the need for many previously maintained data items listed in Figure 5. *Tree Number*, which provides indexing into the parametric tree is no longer required. Linking-type entries related to the format of the output file - *Reference Number (S03)*, *Comment Number (S03)*, *Reference Number (S04)*, *Reference Line Number*, *Comment Number (S05)*, and *Comment Line Number* - are eliminated. Finally, the entry *Measurement Name (S03)* is replaced by the attribute name itself.

At this point, all meaningful data entries of the TERF have been integrated into one comprehensive, encapsulated model.

E. SUFFIX CODING AND THE SUFFIX TABLE

EWIRDB suffix-coded data and the suffix table representation of data comprise the most difficult modeling task in the conceptual design. Suffix coding is incorporated within the EWIRDB to describe the vast array of mode combinations an emitter may exhibit. In effect, suffix-coding links together the parametric values that characterize known or suspected emitter usage. A particular combination of parametric values defines a given mode; suffix coding and suffix table thus provide the means for establishing relationships between parametric values throughout the database. (A comprehensive review of suffix coding and the suffix table is provided in [1].)

Herein lies the complexity in modeling modal relationships. Parametric values are synonymous with attribute values. The attributes whose values describe a given mode are likely interspersed throughout the many classes in the schema. The relationships defined by suffix coding and the suffix table therefore describe associations between attributes -- not classes. An additional complication is the possibility of multiple values (multiple instantiations of the object that contains the attribute) each for a given attribute. Modeling modal (attribute) relationships is difficult because neither the OODM, nor any other data model, explicitly supports such a capability. From a modeling perspective, the problem of representing modal relationships such as those found in the suffix table reduces to problem of representing attribute-to-attribute relationships and attribute-to-attribute combinations.

Upgrading each attribute to a class is an ineffective solution. Related attributes are grouped into classes for the purpose of collectively describing the characteristics of a particular set of objects. The transformation of attribute to class obscures these semantics; each attribute instead becomes a reference within a given class. Moreover, the problem of modeling combinations remains unsolved. There exists no "built-in" OODM mechanism for the purpose of defining combinations of objects, not to mention attributes, throughout a schema.

The process of defining modal combinations, regardless of the modeling tool used, is formidable in the realization that an emitter could likely exhibit hundreds of thousands

of modes. Object-orientation does not appear to simplify this task. Despite its dependence on an artificial labeling system and a non-intuitive table representation, suffix coding has proven to be effective in this combination-oriented modeling. Moreover, it helps to link signal signatures to emitters. At present, I am unable to offer an easier or more viable modeling alternative. The current methodology is therefore incorporated into the object-oriented conceptual design.

The conceptual schema includes a suffix code entry for every attribute throughout the schema; a suffix code entry can be made for every attribute in each instantiation of the object to which the attribute belongs. This provides for the same modeling flexibility as exists in the current model: the binding together of related parameters, the labeling of multiple values for a given attribute, and the inclusion of suffix-coded data within the **SUFFIX TABLE** class of objects (Figures 9, 10, 11) to develop modal combinations. **SUFFIX TABLE** objects would also contain a method, **expand table** (not shown), to return all combinations represented in the suffix-coded data table.

In the object-oriented design, the use of the special suffix codes is no longer useful. The semantics of the special suffix code, ++, used to indicate that a particular parametric value is present in all modes, may be addressed via comment in the **DATA COMMENT** class (Figure 15). The special code, //, applies specifically to the parametric tree structure and indicates that a given value pertains to all modes described within the subtree rooted at the branch containing the special code. Such semantics are implicit in inheritance and aggregation hierarchies, and may be explicitly addressed via comment.

This completes the conceptual design phase. The next stage in the overall design process is the logical design, addressed in Chapter V.

V. A LOGICAL OBJECT-ORIENTED EWIRDB

The O-ODDL native to the M²DBS in the NPS Laboratory for Database Systems Research provides the structure of the logical design presented in this chapter. The O-ODDL, designed and specified in [12], thus provides the means to convert the conceptual database as proposed in Chapter IV into an M²DBS-compatible representation.

Still in its first phase of development, the object-oriented interface to the M²DBS is functional but does not yet support all aspects of the object-oriented paradigm. To date, methods and the aggregation abstraction are not implementable on the M²DBS. Therefore, in the logical design, aggregation will be represented via relationships, and methods will not be addressed.

It is not necessary to address all aspects of the conceptual schema in the logical design to demonstrate the operation of the M²DBS object-oriented interface in handling EWIRDB data. To this end, I address a representative portion of the conceptual schema in developing the logical design. The subsequent implementation of the logical schema on the M²DBS is a continuation of the work in this thesis, and is addressed in [13, unpublished]. The final result of this combined effort will be an on-line object-oriented EWIRDB with which to demonstrate both the utility of the new M²DBS object-oriented interface, and the usefulness of the new object-oriented EWIRDB design.

The O-ODDL logical design constructs are reproduced in Figures 17 through 20. Refer to [12] for a comprehensive discussion of the O-ODDL specification.

In Figure 17, "attribute type" refers to the traditional notion of attribute domain. As described earlier in sections III.A.1.a and b, the domain or type of an attribute may be simple, characterized by literal attribute values, or the type may be complex, characterized by object attribute values. Complex attributes can therefore exhibit an arbitrarily deep nesting of objects. Whereas a simple attribute may be of type "character" or "integer", the type of a complex (or reference) attribute is a class. The class defines the legal attribute values (object values) for the given attribute.

Figures 18 and 19 contain the specifications for the inheritance and covering abstractions. In Figure 20, one-to-many and many-to-many relationships are addressed. One-to-many (1:N) relationships between object classes are represented via the `set_of` construct. `set_of` appears within the class on the “1” side of the relationship; an attribute that references the class on the “1” side of the relationship appears within the class on the “N” side of the relationship. Many-to-many (N:M) relationships are represented with the `set_of` (N side) and `inverse_of` (M side) constructs. One-to-one (1:1) relationships are represented directly through use of reference attributes. The classes in the 1:1 relationship each contain an attribute whose type is that of the class to which it is associated via the 1:1 relationship.

```

Class Class_name{
    attribute_type1 attribute_name1;
    attribute_type2 attribute_name2;
    .
    .
    attribute_typen attribute_namen
};

```

Figure 17. The O-ODDL Class Specification

```

Class Class_name_X1 : inherit Class_name_X{
    attribute_type1 attribute_name1;
    attribute_type2 attribute_name2;
    .
    .
    attribute_typen attribute_namen
};

```

Figure 18. The O-ODDL Inheritance Specification

```

Class Class_name_X1 : cover Class_name_X{
    attribute_type1 attribute_name1;
    attribute_type2 attribute_name2;
    .
    .
    attribute_typen attribute_namen
};

```

Figure 19. The O-ODDL Cover Specification

```

set_of Class_name attribute_name;

inverse_of Class_name.attribute_name attribute_name;

```

Figure 20. The O-ODDL Set_of and Inverse_of Specifications

The logical design incorporates the O-ODDL constructs as outlined in Figures 21 through 29. Because all complex attributes in the object-oriented design of EWIRDB, ultimately lead to objects of type **PARAMETRIC DATA** (section IV.D), the logical design begins with the O-ODDL representation of **PARAMETRIC DATA** in Figure 21. The design then proceeds with the classes **EMITTER** (Figure 22), **KILTING EMITTER** and **KILTING ADMINISTRATIVE DATA** (Figure 23), **ANTENNA** (Figure 24), **SIGNAL** (Figure 25), **RECEIVER** (Figure 26), **WARM** (Figure 27), **SUFFIX TABLE** (Figure 28) and **S&TI EMITTER** and **S&TI ADMINISTRATIVE DATA** (Figure 29).

<pre> class Parametric_Data{ char* Data_Comment Data_Descript set_of Orig_Src_Doc; }; </pre>	<pre> suffix_code; comments; description; references; </pre>
<pre> class Textual_Data : inherit Parametric_Data { char* }; </pre>	<pre> text; </pre>
<pre> class Numeric_Data : inherit Parametric_Data { char* }; </pre>	<pre> units; </pre>
<pre> class Specific_Value : inherit Numeric_Data{ char* }; </pre>	<pre> value; </pre>
<pre> class Value_Range : inherit Numeric_Data{ char* char* }; </pre>	<pre> upper_value; lower_value; </pre>
<pre> class Data_Comment { char* Parametric_Data }; </pre>	<pre> comment_text; parametric_data; </pre>
<pre> class Assess_Comment : inherit Data_Comment { char* }; </pre>	<pre> com_classif; </pre>

**Figure 21. The Logical Design: DDL Implementation of the
PARAMETRIC DATA Class (cont'd into next page)**

<pre> class Data_Descrip { char* Parametric_Data }; </pre>	<pre> last_update; para_data; </pre>
<pre> class Assessed_Data : inherit Data_Descrip{ char* char* char* }; </pre>	<pre> confidence_level; classification; releasability; </pre>
<pre> class Observed_Data : inherit Data_Descrip{ char* char* char* int }; </pre>	<pre> meas_accuracy; meas_acc_units; intel_source; prefer_rating; </pre>
<pre> class Orig_Src_Doc { char* char* inverse_of Parametric_Data.references }; </pre>	<pre> rpt_classif; rpt_release; p_data; </pre>
<pre> class Assessed_Ref: inherit Orig_Src_Doc { char* }; </pre>	<pre> reference_text; </pre>
<pre> class Observed_Ref: inherit Orig_Src_Doc { char* char* char* char* char* }; </pre>	<pre> document_number; document_title; report_date; producer; report_classification; </pre>

Figure 21. (cont'd) The Logical Design: DDL Implementation of the PARAMETRIC DATA Class

```

class Emitter{
    char*                               elnot;
    char*                               color;
    Kilting Emitter                     kilting_data;
    S&TI Emitter                         s_and_ti_data;
};

```

**Figure 22. The Logical Design: DDL Implementation of the
EMITTER Class**

```

class Kilting Emitter{
    char*                               technical_date;
    Kilting_Admin                       k_admin_data;
    set_of Antenna                       antenna_data;
    set_of Signal                        signal_data;
    set_of Receiver                      receiver_data;
    set_of WARM                           war_res_modes;
    Suffix_Table                          modes;
    Parametric_Data                      weapon_system;
    Parametric_Data                      function;
    Parametric_Data                      platform;
    Emitter                              general_data;
};

```

```

class Kilting_Admin{
    char*                               nsa_date;
    char*                               sae_code;
    char*                               date_sig_change;
    char*                               kclassification;
    char*                               kreleasability;
};

```

**Figure 23. The Logical Design: DDL Implementation of the
KILTING EMITTER and KILTING
ADMINISTRATIVE DATA Classes**

class <i>Antenna</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Polarization</i> <i>Radiation_Pattern</i> <i>Kilting_Emitter</i> };	antenna_type; antenna_function; horizontal_dimension; vertical_dimension; ant_polarization; ant_rad_char; kilt_emitter;
class <i>Polarization</i> { <i>Cross_Polarization</i> <i>Antenna</i> };	cross_pol_char; antenna;
class <i>Cross_Polarization</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Polarization</i> };	patt_pk_offset; patt_pk_resp; polarization; };
class <i>Linear_Polarization</i> : inherit <i>Polarization</i> { <i>Parametric_Data</i> };	major_axis_tilt_angle;
class <i>Circ_or_Ellipt_Polarization</i> : inherit <i>Polarization</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> };	sense; axial_ratio;

Figure 24. The Logical Design: DDL Implementation of the ANTENNA Class (cont'd into next page)

class <i>Radiation_Pattern</i> { <i>Parametric_Data</i> <i>Antenna</i> };	antenna_gain; ant;
class <i>Directional_Ant</i> : inherit <i>Radiation_Pattern</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> <i>set_of_Scan</i> <i>set_of_Track</i> };	beamwidth_az; beamwidth_el; first_sidelobe_lvl_az; first_sidelobe_lvl_el; scanning_char; tracking_char;
class <i>Omnidirectional</i> : inherit <i>Radiation_Pattern</i> { <i>Parametric_Data</i> elevation_coverage_angle; };	
class <i>Scan</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Directional_Ant</i> };	sample_avg_time; SNR_threshold; plane_of_scan; dir_antenna;
class <i>Mechanical_Scan</i> : inherit <i>Scan</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> };	type_change; scan_function;
class <i>Circular</i> : inherit <i>Mechanical_Scan</i> { <i>Parametric_Data</i> };	cperiod_limits;

Figure 24. (cont'd) The Logical Design: DDL Implementation of the ANTENNA Class (cont'd into next page)

class Sector : inherit Mechanical_Scan{ <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> };	
class Track{ <i>Parametric_Data</i> <i>Directional_Ant</i> };	sector_type; speriod_limits; sector_width_az; sector_width_el; plane_of_scan; direct_ant;
class Mech_Tracking : inherit Track{ <i>Parametric_Data</i> auto_track_max_rate_az; <i>Parametric_Data</i> auto_track_max_rate_el; };	

Figure 24. (cont'd) The Logical Design: DDL Implementation of the ANTENNA Class

class <i>Signal</i> { <i>Trans_Power</i> <i>Kilting_Emitter</i> };	signal_pwr; k_emitter;
class <i>Trans_Power</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Signal</i> };	line_loss_on_tx; pk_pwr_eff_rad; pk_pwr_at_trans; signal;
class <i>Constant_Power</i> : inherit <i>Transmission_Power</i> { <i>Parametric_Data</i> };	time_to_switch;
class <i>Not_Constant_Power</i> : inherit <i>Transmission_Power</i> { <i>Parametric_Data</i> };	max_rate_of_change;
class <i>Pulsed_RF</i> : inherit <i>Signal</i> { <i>RF_Line_Structure</i> <i>Pulse_Duration</i> <i>PRI/PGRI</i> };	coherence; pulse_length; pulse_groups;
class <i>RF_Line_Structure</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Pulsed_RF</i> };	3_db_spec_width; trans_type; rf_pulse;

Figure 25. The Logical Design: DDL Implementation of the SIGNAL Class (cont'd into next page)

class <i>Pulse_Duration</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Pulsed_RF</i> };	pulse_dur_lim; pd_most_prob; pulse;
class <i>PD_Modulated</i> : inherit <i>Pulse_Duration</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> };	dev_limits; modulation_rate;
class <i>RF_Constant</i> : inherit <i>Pulsed_RF</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> };	rf_limits; most_prob_rf;
class <i>RF_Not_Constant</i> : inherit <i>Pulsed_RF</i> { };	
class <i>Mod_on_Pulse</i> : inherit <i>RF_Not_constant</i> { <i>Parametric_Data</i> };	rf_mod_change;
class <i>PMOP</i> : inherit <i>Mod_on_Pulse</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> };	rf_limits; phase_shift;
class <i>FMOP</i> : inherit <i>Mod_on_Pulse</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> };	fmop_rf_limits; freq_excursion;

Figure 25. (cont'd) The Logical Design: DDL Implementation of the SIGNAL Class (cont'd into next page)

class <i>Pulsed_Agility</i> : inherit <i>RF_Not_constant</i> { <i>Parametric_Data</i> <i>agil_func_corr</i> ; <i>Parametric_Data</i> <i>mod_waveform</i> ; };
class <i>Cont_Agility</i> : inherit <i>Pulsed_Agility</i> { <i>Parametric_Data</i> <i>rf_limits</i> ; };
class <i>Disc_Agility</i> : inherit <i>Pulsed_Agility</i> { <i>Parametric_Data</i> <i>rf_limits</i> ; <i>Parametric_Data</i> <i>no_disc_steps</i> ; };
class <i>PRI</i> { <i>Parametric_Data</i> <i>meas_bandwidth</i> ; <i>Pulsed_RF</i> <i>rf</i> ; };
class <i>Not_Const_PRI</i> : inherit <i>PRI</i> { <i>Parametric_Data</i> <i>modulation_type</i> ; <i>Intvl_Sked</i> <i>interval_cntrl</i> ; };
class <i>Intvl_Sked</i> { <i>Parametric_Data</i> <i>duty_cycle</i> ; <i>set_of Recurrent_Intvl</i> <i>intervals</i> ; };

Figure 25. (cont'd) The Logical Design: DDL Implementation of the SIGNAL Class

class <i>Recurrent_Intvl</i> { <i>Parametric_Data</i> <i>Intvl_Sked</i> };	<i>nbr_dscrete_int</i> ; <i>sked_control</i> ;
class <i>Recur_Intvl_Seq</i> : cover <i>Recurrent_Intvl</i> { <i>Parametric_Data</i> };	<i>sequence_1</i> ;

Figure 25. (cont'd) The Logical Design: DDL Implementation of the SIGNAL Class

class <i>Receiver</i> { <i>Parametric_Data</i> <i>Sig_Proc_Sect</i> <i>A_D_Conv_Sect</i> <i>S&TI_Emitter</i> };	<i>receiver_type</i> ; <i>sig_processor</i> ; <i>a_d_section</i> ; <i>s_emitter</i> ;
class <i>Sig_Proc_Sect</i> { <i>Doppler_Processing</i> <i>Receiver</i> };	<i>dop_proc</i> ; <i>receiver</i> ;
class <i>Doppler_Processing</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Sig_Proc_Sect</i> };	<i>coh_proc_intrvl</i> ; <i>pulses_in_cpi</i> ; <i>processor</i>

Figure 26. The Logical Design: DDL Implementation of the RECEIVER Class (cont'd into next page)

class <i>Multiple_Pulse_Processing</i> { <i>Doppler_Processing</i> };	mp_dop_proc;
class <i>A_D_Conv_Sect</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Receiver</i> };	a_sample_period; conv_trig_meth; rcvr;
class <i>Single_Pulse_Processing</i> { <i>A_D_Convr_Sect</i> <i>Pulse_Compression</i> };	a_d_converter; pulse_compress;
class <i>Pulse_Compression</i> { <i>Parametric_Data</i> <i>Parametric_Data</i> <i>Sig_Pulse_Proc</i> };	type_of_coding; time_band_prod; single_pulse;

Figure 26. The Logical Design: DDL Implementation of the RECEIVER Class

class <i>WARM</i> { char* <i>Kilting_Emitter</i> };	prob_code; kilt_emit;
class <i>Power_ECCM</i> : inherit <i>WARM</i> { set_of <i>Trans_Power</i> };	res_pwr_mode;
class <i>Polar_ECCM</i> : inherit <i>WARM</i> { set_of <i>Polarization</i> };	res_polar_mode;
class <i>Ant_Scan_ECCM</i> : inherit <i>WARM</i> { set_of <i>Scan</i> };	res_scan_mode;
class <i>Sig_Shape_ECCM</i> : inherit <i>WARM</i> { set_of <i>Pulse_Duration</i> };	res_pd_mode;
class <i>RF_ECCM</i> : inherit <i>WARM</i> { set_of <i>Pulsed_RF</i> };	res_rf_mode;

Figure 27. The Logical Design: DDL Implementation of the WARM Class


```

class Suffix_Table{
    char*
                                modematrix;
};

```

Figure 28. The Logical Design: DDL Implementation of the SUFFIX TABLE Class

```

class S&TI_Emitter{

    S&TI_Admin
    set_of Antenna
    set_of Signal
    set_of Receiver
    set_of WARM
    Suffix_Table
    Parametric_Data
    Parametric_Data
    Parametric_Data
    Emitter

    k_admin_data;
    antenna_data;
    signal_data;
    receiver_data;
    war_res_modes;
    modes;
    weapon_system;
    function;
    platform;
    general_data;
};

class S&TI_Admin {
    char*
    char*
    char*
    char*

    s&ti_code;
    mult_src_review;
    sclassification;
    sreleasability;
};

```

Figure 29. The Logical Design: DDL Implementation of the S&TI EMITTER and S&TI ADMINISTRATIVE DATA Classes

The effect of the object-oriented logical design is profound. Now, all available data for a given emitter, both technical and administrative, is contained within an object of the class **EMITTER**. This effect is achieved via the nesting of objects within the framework of relationships, inheritances, and a covering.

EMITTER contains complex (reference) attributes (object values) of type **PARAMETRIC DATA**, and also contains references to source-specific emitter data objects of type **KILTING EMITTER** and **S&TI EMITTER**. **KILTING EMITTER** and **S&TI EMITTER** objects likewise contain attributes of type **PARAMETRIC DATA**, and attributes that reference analogous administrative data objects. These administrative data objects contain simple-valued, source-specific attributes corresponding to S00, S01, and S02 record data. The **KILTING EMITTER** and **S&TI EMITTER** objects additionally encapsulate antenna data, signal data, receiver data, WARM data, and suffix table objects. (Suffix table objects correspond to S05 record data). The attributes within each of these objects, in turn, are either of type **PARAMETRIC DATA**, or exhibit a nesting of objects that ultimately lead to attributes of type **PARAMETRIC DATA**.

Finally, attributes of type **PARAMETRIC DATA** exhibit a nesting of objects that leads to simple parametric values and simple parametric-value-related administrative data. All such information corresponds to S03 record data.

The overall result is a cohesive, encapsulated, and comprehensive logical schema of EWIRDB data.

VI. CONCLUSION

The EWIRDB is a vitally important instrument of EW and EW research and development, containing up-to-date and mission-critical performance data on the EW systems of friendly and hostile forces. Its utilization in the areas of battle planning and EW research helps to shape the outcome of war. The usefulness of the EWIRDB, however, is hampered by its cumbersome data model, the basis of which is an inherently arbitrary parametric tree structure. The inconsistencies that exist among the data as modeled in the parametric tree and the data as addressed in the record-based output file further obscure the intended semantics. The overall data representation is non-intuitive, disjoint, and difficult to comprehend. The burden of data interpretation is transferred to the user, and the user must deal at length with formatting and coding issues.

In this thesis, I have proposed an alternative and improved representation of EWIRDB data. The design effort was centered on the development of a legitimate conceptual design, followed by development of a logical design suitable for implementation on the M²DBS in the NPS Laboratory for Database Systems Research. The conceptual and logical designs are the first two phases in the overall database design and implementation process.

The conceptual design has yielded a conceptual schema that captured the nature of a representative portion of EWIRDB data in a way that closely paralleled the user's perception of the data. The basis of the conceptual design was the OODM, a powerful modeling tool that enables the designer to reduce the semantic mismatch between real-world entities and their database representations. The OODM incorporates the concepts of objects, encapsulation, object classes, instantiation and classification, generalization and specialization, aggregation, and covering to achieve this end. The object-oriented conceptual design has captured both the technical and administrative semantics of EW data to a degree not previously achievable. This was the realization of the primary objective of the thesis.

In the logical design, I have mapped the object-oriented conceptual schema to the object-oriented data model of the M²DBS. The mapping is accomplished via the O-ODDL native to the M²DBS. The resulting O-ODDL statements constitute the logical schema; they are an M²DBS-readable specification of the conceptual schema. The O-ODDL provided for an arbitrarily deep nesting of objects within a framework of relationships, inheritance, and covering. The semantics of the data have been preserved in the mapping; when implemented on the M²DBS, these semantics will be supported by the M²DBS. This is a huge benefit - the database user is thereafter relieved of the responsibility of data translation and interpretation. Although it does not yet support methods or aggregation, the O-ODDL provides for an intuitive, cohesive, and nested implementation of technical and administrative data. Therefore, the implementation is much improved over the complex record-based format that currently exists.

The logical design portion of this work provides input for the subsequent use and evaluation of the object-oriented interface to the M²DBS, and in this regard satisfies the secondary objective of the thesis. In due course, the logical schema will be implemented on the M²DBS to produce an on-line object-oriented EWIRDB with which to demonstrate both the utility of the new M²DBS object-oriented interface and the usefulness of the new object-oriented EWIRDB design.

Object-orientation did not appear to simplify the formidable task of modeling emitter mode combinations, currently represented through use of suffix codes and suffix tables. For this reason, I retained the suffix code-suffix table system in the designs presented in this thesis. Consequently, the use of this system complicates the implementation of the database. In the object-oriented approach, however, a reliance on external software to interface with suffix tables is unnecessary. Such manipulation may be achieved internal to the DBMS via methods. A true modeling solution may depend on the development of a data model that provides the flexibility to address attribute-to-attribute relationships and combinations.

Overall, the conceptual and logical designs developed in this thesis support and confirm the object-oriented approach as a viable solution to the modeling inadequacies of the present EWIRDB.

LIST OF REFERENCES

- [1] National Air Intelligence Center, *Electronic Warfare Integrated Reprogramming Database (EWIRDB) Guide*, Volume 1, April 1994.
- [2] Elmasri, R., and Navathe, S., *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, Inc., 1994.
- [3] Kim, W., "Object-Oriented Databases: Definition and Research Directions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, no. 3, September 1990.
- [4] Cattell, R., *Object Data Management - Object-Oriented and Extended Relational Database Systems*, Addison-Wesley Publishing Company, Inc., 1994.
- [5] Bertino, E., Martino, L., "Object-Oriented Database Management Systems: Concepts and Issues," *Computer*, April 1991.
- [6] Hsiao, D., "The Object-Oriented Database Management: A Tutorial On It's Fundamentals," *Proceedings of the Second Far-East Workshop on Future Database Systems*, Kyoto, Japan, April 1992.
- [7] Hsiao, D., "Interoperable and Multidatabase Solutions for Heterogeneous Databases and Transactions," a speech delivered at ACM CSC 1995, Nashville, Tennessee, March 1995.
- [8] National Air Intelligence Center, *Pulsed/CW Tree Measurement Guide Table*, July 1992.
- [9] National Air Intelligence Center, *RPA Measurement Guide Table*, July 1992.
- [10] Stimson, G., *Introduction to Airborne Radar*, Hughes Aircraft Company, 1983.
- [11] Frieden, R., *Principles of Naval Weapons Systems*, United States Naval Institute, 1985.
- [12] Badgett, R., *Design and Specification of an Object-Oriented Data Definition Language*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1995.
- [13] McKenna, T., Lee, J., *The Object-Oriented Database and Processing of Electronic Warfare Data*, Master's Thesis, Naval Postgraduate School, Monterey, California, to be published March 1996.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
 8725 John J. Kingman Rd., STE 0944
 Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library, Code 13 2
 Naval Postgraduate School
 Monterey, California 93943-5101

3. Sharon Cain 1
 NAIC/SCDD
 4115 Hebble Creek Rd
 Wright-Patterson AFB, Ohio 45433

4. Chairman, Code CS 1
 Computer Science Department
 Naval Postgraduate School
 Monterey, California 93943

5. Dr. David K. Hsiao, Code CS/HS 1
 Computer Science Department
 Naval Postgraduate School
 Monterey, California 93943

6. Dr C. Thomas Wu, Code CS/KA 1
 Computer Science Department
 Naval Postgraduate School
 Monterey, California 93943

7. LT Kevin M. Coyne 1
 203 Colmar Rd
 Seaside, California 93955